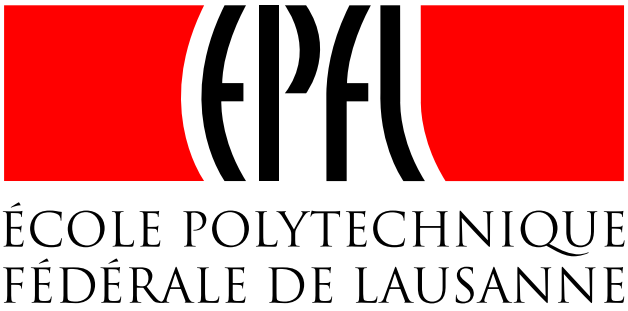

Scheduling with Multiprocessors: A rounding network algorithm with a constant error

Master Thesis

Oliver Ittig
Graduate Masters Student
Mathématique, EPFL

Supervisors:

Prof. Wieslaw Kubiak
Faculty of Buisness Administration
Memorial University of Newfoundland

Prof. Dominique de Werra
Departments of Mathematics
Swiss Federal Institute of Technology

Winter 2004/2005

Acknowledgments

It was an enjoyable time here at the MUN and in St. John's. In more then one way it was a very instructional time. It was very impressing to be able to work four months on one subject, to have the time to acquire a subject in such a profound way and also to do research at another university in a country with different people, language and culture. It was also fascinating to see how the interest and the motivation about the work increased by learning more about the topic. For me personally, it was a great success, because I thought always that I am much more a person who is more interested in applications then in proofing. But I took much pleasure in demonstrating the obtained results. It was a new personal keen insight.

At last I want to thank Prof. Kubiak many times for all his support, ideas, suggestions, but also for his extracurricular support. I want thank Prof. de Werra for the organization of the exchange and his support form the side of the EPFL and of course I want to thank my family for the support during my entire studies and for everything that they enabled me to do.

Contents

List of Figures

Chapter 1

Introduction

For many kinds of scheduling open shops are used. In this thesis we consider not only individual processors, as many examples require the use of a multiprocessor. For example, on a car assembly line a multiprocessor could be a robot with more then one arm. All arms work simultaneously on a car, putting simultaneously the windshield, the rear window and the driving mirrors. Also in computer systems we use multiprocessors as in computing circuits or printing devices. In this thesis, we focus on multiprocessors in timetabling. The most common model is the class-teacher model. There are m classes $C_1, \ldots, C_m$ and n teachers $T_1, \ldots, T_n$. Each class C_h has b_{jh} lectures given by teacher T_j. Moreover, each class is not to be taught by more then one teacher at a time, and no teacher may teach more then one class at at time. The university timetable model, which is of our main interest, represents the extended version including the multiprocessors. The extension is a lecture to a group of classes, for example a basic mathematics course offered for all biology, chemistry and earth science students in an undergraduate program. Then, no teacher must teach more then one class or group at a time. Imagine the practical industrial examples where multiple robots work simultaneously on the same job, or computer networks where multiple processors execute the same task at the same time. Herein we show existing resolutions and results for such examples with zero or two multiprocessors in Chapter 3. Moreover, the formulation of problems like these with one, three or more multiprocessors will be illustrated and discussed in Chapter 4. The formulation assumes a strict partition of the multiprocessors where no individual processor occurs in more then one multiprocessor.

An interesting problem to study is the difference between fractional and integer preemptions. If we work with machines, processors, as for example in computer systems, etc. we might use fractional preemptions, as we can interrupt any process at any fractional time point. But if we consider schedules like school timetables, then we have to consider integer preemptions. To compute schedules with fractional preemptions, we use Linear Programming (LP). But to find a schedule with integer preemptions for more then three multiprocessors becomes

$\mathcal{NP}$-hard (see [1]). Therefore, we consider a rounding network in Chapter 5; the network uses the optimal LP solution for the fractional case and gives us an instruction how to round it up. Interestingly, the resulting makespan is a constant distance form the fractional makespan. We discussed the rounding network also for the special case of the professor-lecturer model (see Section 5.5) shown by Asratian and de Werra in [1].

We also introduce a theoretical graph algorithm in Chapter 6, since finding an optimal makespan for such a timetable with integer preemptions is equivalent to finding a proper edge coloring for a bipartite graph, i.e. with no multiprocessor, or a proper edge coloring in a hypergraph, i.e. with multiprocessors. This subject has been discussed in [2]. However, to decide if a timetable with makespan t exits, is $\mathcal{NP}$-complete in a hypergraph.

Finally we compare in Chapter 7 our new methods with the approximation algorithms of Asratian and de Werra in [1], but first we start with a short introduction to the terminology in the next chapter.

Chapter 2

Definitions and terminology

In this chapter we introduce the terminology used in this thesis with respect to the graph theory, linear programming and scheduling with open shops.

2.1 Graph theory

For more information about the graph theory see Berge [3].

2.1.1 Basic concepts

A *graph* G is a pair (V, E) where V is a set of *vertices* and E is a set of *edges*. The vertices are points that are linked by the edges. If we have an edge $e \in E$ joining two vertices $u, v \in V$, then we write $e = \{u, v\}$. An edge is *incident* with a vertex v if one of its ends is v. Two vertices are called *adjacent* if they are joined by an edge. Two edges are called *adjacent*, if they are incident with the same vertex. The degree $d_G(v)$ of a vertex v is the number of edges incident to v. The maximum degree of G is denoted as $\Delta(G)$.

If the graph G has more then one edge between u and v, we call G a *multigraph*. An edge of the form $\{u, u\}$ is called a *loop*.

A multigraph is called a *simple graph*, if 1) it has no loops and 2) no more than one edge joins any two of its vertices.

A multigraph is called a *bipartite graph*, if the vertex set V can be partitioned into two sets V_1 and V_2 in such a way that no two vertices from the same set are adjacent.

A *directed graph* is a pair (V, A) where V is a set of vertices, and A is a set of *arcs*. An arc is an edge $\{u, v\}$ with a direction and will be written as (u, v). If we write (u, v) then u is the *origin* and v is the *endpoint*.

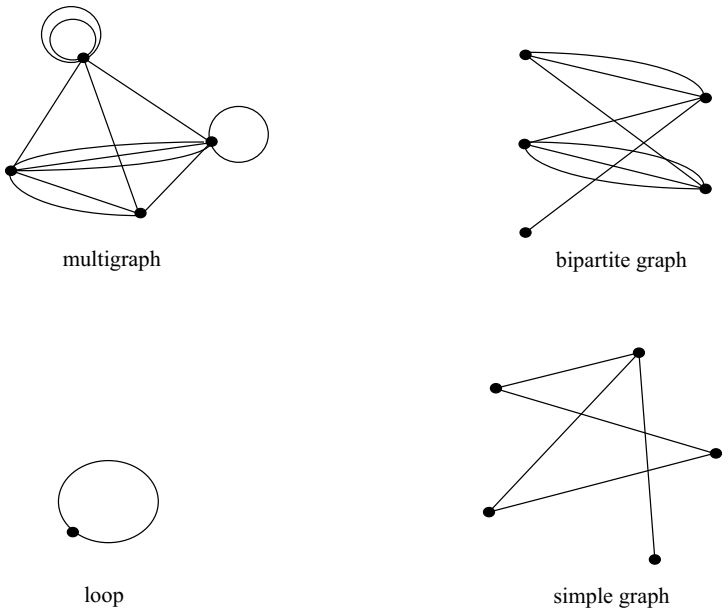

Figure 2.1: Examples of different graphs.

2.1.2 Matching

Given a simple graph G, a *matching* is a set of edges M such that no two edges of M are adjacent. The most common problem is to find a matching with a maximal cardinality $|M|$.

A vertex v is called *saturated* by a matching M if an edge of M is incident to v. Furthermore a *perfect matching* is a matching that saturates all vertices of G. A perfect matching is a maximum matching cardinality.

The edges $e \in M$ are drawn as thick lines in Figure 2.2.

2.1.3 Edge coloring

A *k-edge coloring* $\mathcal{C}$ of a graph G is an assignment of k colors to the edges of G. We assume the graph G has no loop. We call such an assignment *proper* if no pair of edges with a common end is assigned the same color. For a proper k-coloring of G we have a partition E into k subsets $E_1, E_2, \ldots, E_k$, where every subset E_i is a matching.

The *chromatic index* $q(G)$ of a graph G is defined to be the smallest number of colors needed to color the edges of G so that no two adjacent edges have the same color. A *q-coloring of the edges* is defined to be a partition of the edge set divided into q subsets that are matchings, thus

$$q(G) \geq \max_{v \in V} d_G(v) = \Delta(G).$$

In Figure 2.3, we have a 3-coloring of the edges for a simple graph. The Example

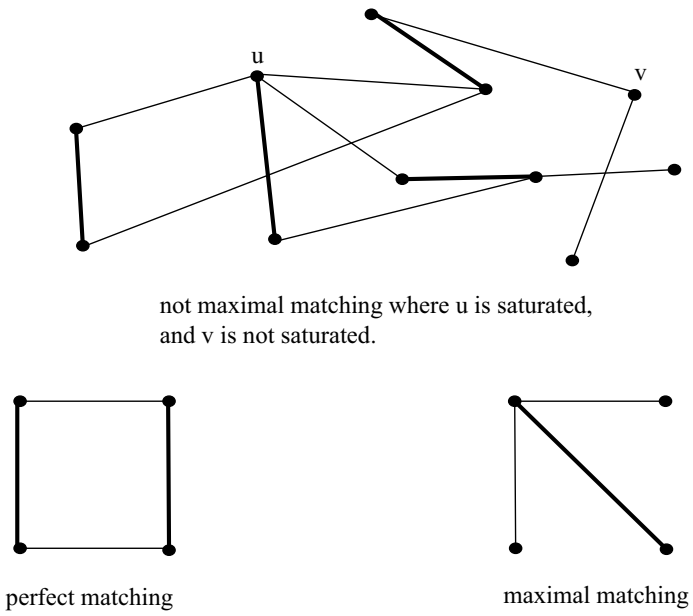

Figure 2.2: Examples of matchings.

is given by Berge [3]. In Figure 2.4 we have a 3-coloring of the edges for a bipartite graph.

Remark 2.1. *The chromatic index $q(G)$ should not be mistaken for the chromatic number $\chi(G)$ which is the smallest number of colors needed to color the vertices.*

Theorem 2.1. *A k-coloring of a bipartite multigraph G exists if and only if $k \geq \Delta(G)$.*

Theorem 2.2 (König's Coloring Theorem)**.** *For any bipartite graph G, we have $q(G) = \Delta(G)$.*

A *V-sequential edge coloring* is an edge coloring of G such that for each node $v \in V$ the edges adjacent to v are assigned colors $1, 2, \ldots, d_G(v)$.

A *interval edge coloring* is introduced by Asratian, Denley and Häggkvist in [2]. Such a interval edge coloring will be considered only for bipartite multigraphs. There we color every edge, but in such a way that all edges with a common vertex will be colored with consecutive colors. If we have two vertices w, v, that are joined by 3 edges, then we color these edges with the colors $k, k+1$ and $k+2$. In Figure 2.5, we have consecutive colors on the left hand. In Figure 2.6, we have a consecutive edge coloring on both sides, i.e. the interval edge coloring.

2.1.4 Hypergraphs

This subsection about hypergraphs is taken from Berge [3]. We have $X = \{x_1, x_2, \ldots, x_n\}$, where X is the set of the vertices, and $\mathcal{E} = (E_i | i \in I)$ is a family of subsets of X. The family $\mathcal{E}$ is said to be a *hypergraph on* X if

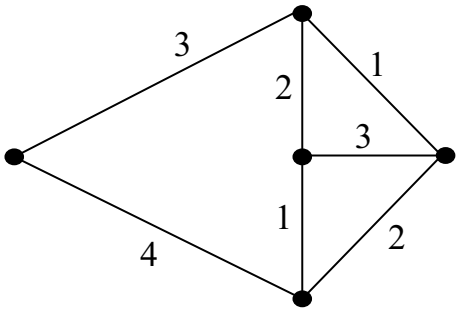

Figure 2.3: Example of edge coloring of a simple graph.

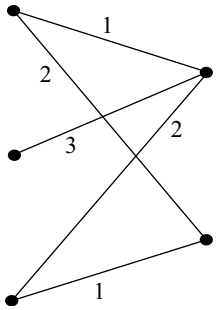

Figure 2.4: Example of edge coloring of a bipartite graph.

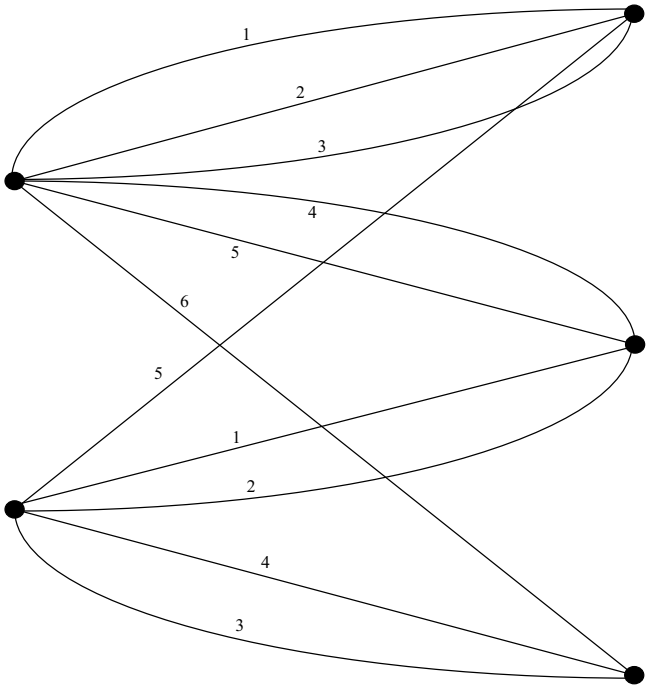

Figure 2.5: Consecutive edge coloring on the left side.

1. $E_i \neq \emptyset,\ (i \in I)$;

2. $\cup_{i \in I} E_i = X$;

Then a *hypergraph* is the pair $H = (X, \mathcal{E})$. We call $x_1, x_2, \ldots, x_n$ the *vertices* and the sets $E_1, E_2, \ldots, E_m$ are denoted as the *edges*. If we have a hyperedge E_i with $|E_i| > 2$, we draw a curve encircling all the vertices in E_i. If $|E_i| = 2$, we have an edge. If $|E_i| = 1$, we have a loop.

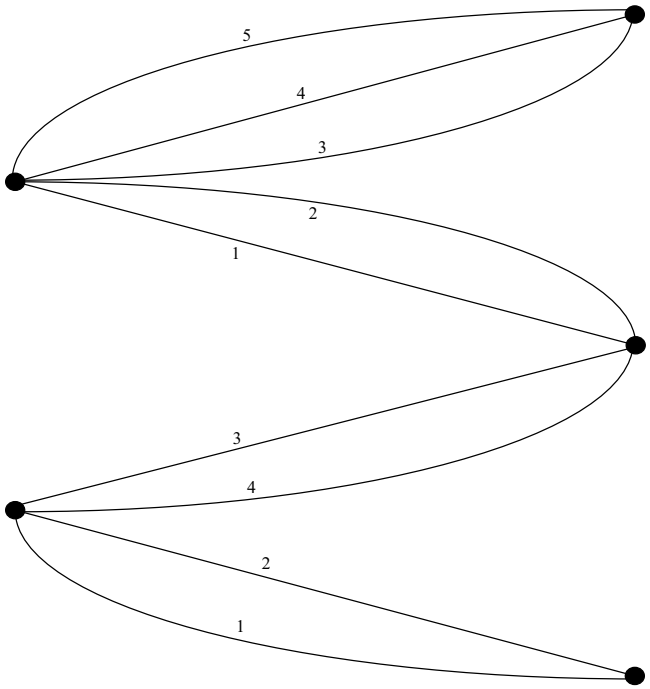

Figure 2.6: Consecutive edge coloring on both sides.

A hypergraph is *simple* if all edges E_i are distinct. If all $|E_i| \leq 2$ then we call a hypergraph simply a hypergraph.

Two vertices are *adjacent*, if they belong to the same hyperedge E_i. Two hyperedges are called adjacent, if their intersection is not empty.

In Figure 2.7 we have a hypergraph with $|X| = 8$ and $|I| = 6$, meaning we have six subsets E_i. In this hypergraph we have a loop, given by E_6. The edge E_1 represents the adjacency for the vertices x_3, x_4 and x_5. The edges E_3, E_4 and E_6 are adjacent. Furthermore, the given hypergraph is simple, because we have $E_i \neq E_j,\ \forall i, j \in I$ and $i \neq j$.

The dual hypergraph

For each hypergraph $H = (X; E_1, E_2, \ldots, E_m)$ there exists a *dual hypergraph* $H^* = (E; X_1, X_2, \ldots, X_n)$ of H, whose vertices are points $e_1, e_2, \ldots, e_m$ (represents $E_1, E_2, \ldots, E_m$) and whose edges are sets $X_1, X_2, \ldots, X_n$ (representing $x_1, x_2, \ldots, x_n$ respectively), where $\forall j$ we have

$$X_j = \{e_i | i \leq m, x_j \in E_i\}.$$

There we have $X_j \neq \emptyset$ and $\cup_j X_j = E$, and H^* is a hypergraph.

In Figure 2.8 we see the dual hypergraph of the hypergraph in Figure 2.7. Each vertex x_j of H is transformed into a subset X_j of H^* and the analog each edge E_i is transformed into a vertex e_i.

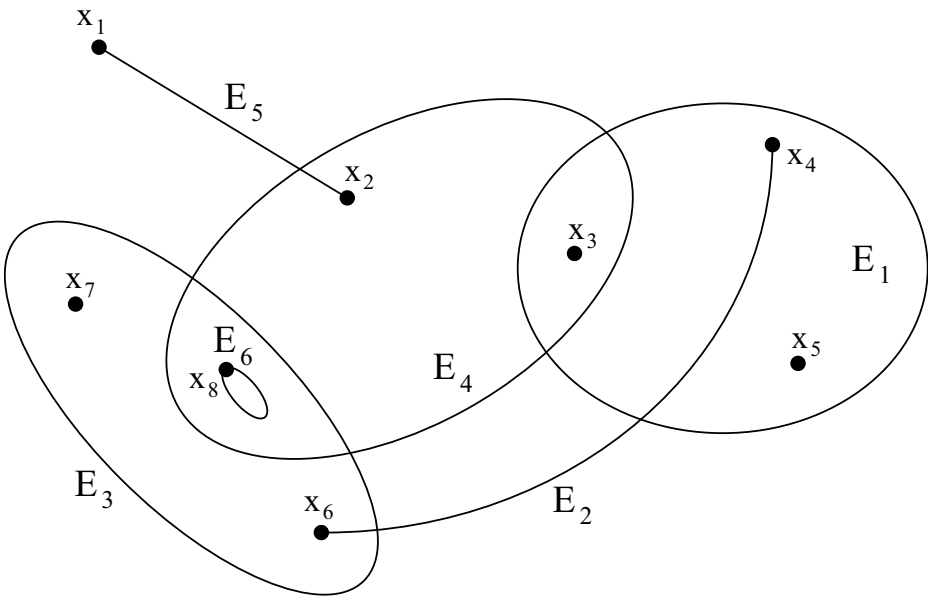

Figure 2.7: Example of a hypergraph.

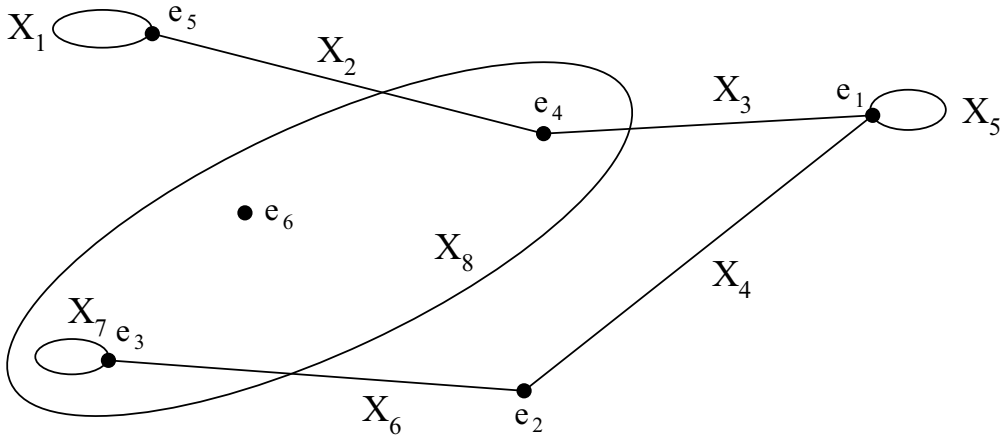

Figure 2.8: The corresponding dual hypergraph to the hypergraph in Figure 2.7.

Hypergraph coloring

The references about hypergraph coloring are form Berge in [3] and from I. Dinur, O. Regev and C. Smyth in [9].

The *chromatic number* $\chi(H)$ is the smallest number of colors needed to color the vertices of H in such a way that no edge E_i ($|E_i| > 1$) has all its vertices colored with a different color. A *t-coloring* is a partition of X in t sets, where each set is assigned a color. A hypergraph for which there exists a t-coloring is called *t-colorable*. Further, we call the hypergraph *t-chromatic*, if $t = \chi(H)$. If $T \geq \chi(H)$, then H is *T-colorable*. If all vertices joining the same edge have the

same color, then we call the edge *monochromatic*.

The *degree* of a vertex x, denoted as $d_H(x)$, is the number of edges which include x. If you look at hypergraph H in Figure 2.7, $d_H(x_1) = 1, d_H(x_2) = 2, d_H(x_4) = 2, d_H(x_8) = 3$.

Theorem 2.3. *If H is a hypergraph of maximum degree $d_{\max}$, then*

$$\chi(H) \leq d_{\max} + 1.$$

The proof is given in Berge [3].

A *t-edge coloring* for the hypergraph H is an assignment of t colors to each edge in such a way that two adjacent edges never have the same color. If t is minimal then we called the edge coloring *proper* and $t = q(H)$. If all edges, attached at a vertex have the same color, the vertex is called *monochromatic*.

The *rank of hypergraph H* is written as $r(X)$.

$$r(X) = \max_i |X \cap E_i|.$$

If $E_i = r(X)$ for each i, then H is a *uniform hypergraph of rank $r(X)$*, also called *r-uniform*. For example, each simple graph is a uniform hypergraph of rank 2.

Remark 2.2. *We introduced the dual hypergraph in the sense of switching between vertex and edge coloring, because most of the recent literature is written about vertex coloring. So it is never a problem to find a proper edge coloring by an algorithm, because we can change between edge coloring on H or vertex coloring on H^*. The transformation $H \mapsto H^*$ or reverse is in $O(|X| + \sum_i |E_i|)$, because all necessary information is saved in the given hypergraph H or respectably in H^*.*

2.2 Linear Programming

A *linear program* is an optimization problem of the form:

$$\begin{aligned} \min z(\overrightarrow{x}) &= c_1x_1 + c_2x_2 + \ldots + c_nx_n \\ \text{s.t.} & \\ & a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ & a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ & \vdots \\ & a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n = b_m \\ & x_i \geq 0, \qquad \forall i = 1, \ldots, n \end{aligned}$$

A non-negative vector $\overrightarrow{x} = (x_1, \ldots, x_n)$ satisfying the equalities of this linear problem is called a *feasible solution*. The aim of this problem is to find an x in such a way the $z(x)$ is optimal, in our case minimal. The number of m constraints does not have to be equal to n.

The linear program can also be given in a matrix form, where the a_{ij} are the entries in a matrix $A_{m \times n}$. In a similar way, we transform the $b_1, \ldots, b_m$ in a vector $\overrightarrow{b}$ and the $c_1, \ldots, c_n$ into $\overrightarrow{c}$. With this notation the linear program looks like

$$\begin{aligned} \min z(\overrightarrow{x}) = \overrightarrow{c} \cdot \overrightarrow{x} & \\ \text{s.t.} \quad A \cdot x \leq \overrightarrow{b} & \\ \overrightarrow{x} \geq \overrightarrow{0} & \end{aligned}$$

A well known method to solve the linear problem is the Simplex Algorithm, see for more information de Werra, Liblieng and Hêche in [16]. It is an iterative procedure which finds an optimal solution or detects infeasibility or unboundedness after a finite number of steps.

An integer linear problem means all variables x_i are restricted to integers.

2.3 Scheduling with open shops

2.3.1 Preliminary definitions

The jobs of a set $\mathcal{J}$, written as $\{J_1, J_2, \ldots, J_n\}$, are processed on m processors. For each job of the set $\mathcal{J}$, a processing order is not given in advance and may be different for different jobs. The processing times for job J_j on processors P_h are given. If such a processing time for J_j is equal to zero on P_h then J_j is not processed on P_h. At any time, each job is processed on at most one processor, and each processor executes no more than one job at a time. Such processing systems are called *open shops*.

An *individual operation* O_{jh} is the performing of job J_j on processor P_h. The operation takes b_{jh} time units, the b_{jh} is called the *processing time* of the operation O_{jh}. The processing times can also be given in a matrix $B_{n \times m}$, where $(B_{jh}) = b_{jh}$.

A *multiprocessor* is a set G_l of processors. A group operation $\hat{O}_{jl}$ requires all processors in G_l simultaneously during a_{jl} time units. The matrix B, the a_{jl} can be given in the matrix $A_{n \times p}$, where $(A_{jl}) = a_{jl}$.

Remark 2.3. *In the entire thesis we always assume integral processing times.*

A *schedule* S allocates one or more time intervals on processors to each job. Moreover, we have the following rules to obey:

- each entry of S is either one of the members of the set $\{J_1, J_2, \ldots, J_m\}$ or is empty;
- the job J_j occurs precisely b_{jh} times in the ith row of S for $j = 1, 2, \ldots, m$;
- at any time point in S all symbols are different;

The definition is based on the definition of a timetable by Asratian and de Werra in [1].

Job J_j completes at C_j where C_j is the earliest time point after that the job J_j does not occur anymore in S. The schedule length, called *makespan*, is defined as $C_{\max} = \max(C_j)$. A schedule for which the makespan is at its minimum will be called optimal.

A *preemption* of a job or an operation means that a process may be interrupted and resumed at a later time. A job or an operation may be interrupted any number of times. Theoretically it is possible to have an infinite number of preemptions, but in practice it makes no sense (eg. can you imagine a painter who takes infinite number of smoke breaks). In this way, we always assume a finite number of preemptions. Also, we assume the interruptions and resumptions do not incur additional cost.

Example 2.1. *In Figure 2.9 the first box defines all processing times for the individual operations. For example, job J_1 takes 1 time unit on processor P_1, two time units on processor P_2 and two time units on multiprocessor G_1. Here in this Example $G_1 = \{P_1, P_2\}$ and $G_2 = \{P_3, P_4\}$. Then in Figure 2.10 we can see that the job J_1 is processed in the first time interval on processor P_2. In the second interval J_1 is taken on multiprocessor G_1, in the third interval on P_1, in the fourth interval on G_1 and in the last interval on processor P_2.*

	P_1	P_2	P_3	P_4
J_1	1	2	0	0
J_2	0	1	2	1
J_3	1	0	2	1

G_1 G_2

G_1	G_2
2	0
0	0
0	1

Figure 2.9: An open shop instance with 4 processors and 2 multiprocessors.

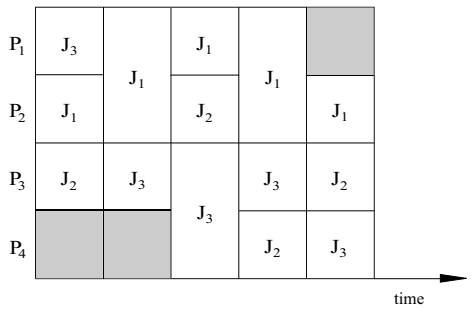

Figure 2.10: A possible schedule with an optimal makespan for the instance in Figure 2.9.

Example 2.2. *The matrixes A and B given here correspond to the Example 2.1. The matrix A represents the processing times for the group operations, where* $(A_{jl}) = a_{jl}$. *The same is valid for the matrix B, where* $(B_{jh}) = b_{jh}$.

$$A = \begin{pmatrix} 1 & 2 & 0 & 0 \\ 0 & 1 & 2 & 1 \\ 1 & 0 & 2 & 1 \end{pmatrix}$$

$$B = \begin{pmatrix} 2 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}$$

Figure 2.11: The matrixes of the individual and group operations of Figure 2.9.

2.3.2 The open shop without multiprocessors

We have a collection $\mathcal{P}$ of m *processors* $P_1, \ldots, P_m$. Provided with these processors is a set $\mathcal{J}$ of n *jobs* $J_1, \ldots, J_n$. Every job J_j takes at most m *operations* $O_{j1}, \ldots, O_{jm}$, where O_{jh} requires processor P_h for processing. The *processing time* of O_{jh} is a non-negative integer number b_{jh}. If $b_{jh} = 0$, then J_j is not processed on processor P_h. We assume a processor could never process two operations of the same job simultaneously. We call O_{jh} individual operations.

We confine the discussion to the preemptive model, so the processing of operations can be interrupted and resumed at a later moment in time. When an operation of a job is interrupted then another operation of the same job may processed before the same operation is resumed. It is assumed that an interruption or a resumptions requires no time and incurs no cost. So we distingish between two models:

1. (PF) : the fractional model;
2. (PI) : the integral model;

In the fractional model preemptions may occur at fractional time points and in the integral model only at integral time points. In either case a *schedule*

specifies for each operation a set of time intervals where the operations must be processed continuously. A schedule is *fractional* (*integral*) if preemptions of processing occur at fractional (only at integral) time points.

Example 2.3. *An Example of a fractional model: Consider a group of three painters (three jobs) who have to paint four walls (four processors) in a house. Any painter can change at any time the wall on which he had work on, and continue the work on an other wall.*

An Example of a integral model: consider a school with 7 teachers (7 jobs) who have to teach 5 classes (5 processors). Here, the teachers always give entire lessons and every lesson takes 45 minutes. So the preemptions, the teacher change the class / the classroom, takes integral time units, here 45 minutes.

2.3.3 The generalized model with p multiprocessors

We have a collection $\mathcal{P}$ of m *processors* $P_1, \ldots, P_m$. Provided with these processors is a set $\mathcal{J}$ of n *jobs* $J_1, \ldots, J_n$. Every job J_j takes at most m *individual operations* $O_{j1}, \ldots, O_{jm}$, where O_{jh} requires processor P_h for processing. The *processing time* of O_{jh} is a non-negative integer number b_{jh}. If $b_{jh} = 0$, then J_j is not performed on processor P_h.

Now we extend the model with the introduction of p multiprocessors. The set $\mathcal{P}$ will be partitioned into p groups, denoted as G_l $\forall l = 1, \ldots, p$. We have $G_1 = \{P_1, \ldots, P_{s_1}\}, G_2 = \{P_{s_1+1}, \ldots, P_{s_2}\}, \ldots, G_{p-1} = \{P_{s_{p-2}+1}, \ldots, P_{s_{p-1}}\}$ and $G_p = \{P_{s_{p-1}+1}, \ldots, P_m\}$, where $1 < s_1 < s_2 < \ldots < s_{p-1} < s_p < m$. Every job J_j takes at most p *group operations* $\hat{O}_{j1}, \ldots, \hat{O}_{jp}$, where $\hat{O}_{jl}$ requires multiprocessor G_l. The *processing time* of $\hat{O}_{jl}$ is a non-negative integer number a_{jl}. If $a_{jl} = 0$, then J_j is not performed on multiprocessor G_l.

We assume a processor (respectively a multiprocessor) could never process two individual operations (respectively two group operations) of the same job simultaneously.

The preemptions, either fractional or integral, can also occur here in the generalized case with multiprocessors.

Remark 2.4. *In this thesis, we always assume a consecutively ordered partition with respect to the multiprocessors. It means we always have consecutive multiprocessors. This order is always possible to obtain. Let us take a look at following Example: we have four processors and two multiprocessors* $G_1 = \{P_1, P_3\}$ *and* $G_2 = \{P_2, P_4\}$. *Now, we can reorder* $P_3 \mapsto P_2^*$ *and* $P_2 \mapsto P_3^*$. *Then we obtain the consecutively reordered multiprocessors* $G_1 = \{P_1, P_2^*\}$ *and* $G_2 = \{P_3^*, P_4\}$. *Of course we have to reindex the corresponding processing times* b_{jh} *and* a_{jl} *too.*

2.4 Network flows

The notation about network flows is based on the terminology from de Werra, Liebling and Hêche in [16] and also from Lawler in [14]. A *network* is a triple, $R = (X, U, C)$, with a *source* s and a *sink* t. X is the set of the vertices, U is the set of arcs (i, j) and C is the set of *capacities* c_{ij}. A capacity c_{ij} is associated with each arc (i, j). Corresponding to the maximal flow problem, we try to transport a maximal quantity, a maximal *flow* v, from the source s to the sink t. In each arc (i, j) we have a quantity x_{ij} which represents the flow in this arc limited by $0 \leq x_{ij} \leq c_{ij}$. The network is a digraph, so we can only transport the quantities x_{ij} from vertex i to vertex j.

We have a *conservation law* at each vertex different from s and t. It says the quantity going out is equal to the quantity arriving at each vertex i. If the searched quantity is v, then the maximal flow problem looks like

$$\begin{aligned} &\max v \\ &\text{s.t.} \quad \textstyle\sum_j x_{ji} - \sum_j x_{ij} = \begin{cases} -v, & i = s \\ 0, & i \neq s, t \\ v, & i = t \end{cases} \\ &0 \leq x_{ij} \leq c_{ij}, \ \forall (i, j) \in U \end{aligned} \tag{2.1}$$

Any set of quantities x_{ij} satisfying (2.1) is a *feasible flow* or simply a *flow* with value v. This problem is a linear program, where v and all x_{ij} are the unknown variables.

We extend the definition of the network flow by adding an lower bound for the quantities x_{ij}. So a network is given as $R = (X, U, C, L)$ and we change the constraint: $l_{ij} \leq x_{ij} \leq c_{ij}$ so the problem looks now like

$$\begin{aligned} &\max v \\ &\text{s.t.} \quad \textstyle\sum_j x_{ji} - \sum_i x_{ij} = \begin{cases} -v, & i = s \\ 0, & i \neq s, t \\ v, & i = t \end{cases} \\ &l_{ij} \leq x_{ij} \leq c_{ij}, \ \forall (i, j) \in U \end{aligned} \tag{2.2}$$

Let us take an undirected path P in R. An arc in P is called *forward*, if it is directed from s to t and *backward* otherwise. An *augmenting path* is a path, where we have $x_{ij} < c_{ij}$ for each forward arc and $x_{ij} > l_{ij}$ for each backward arc in P.

Example 2.4. *Let us consider a feasible flow in a given network in Figure 2.12 from Lawler [14]. There each arc (i, j) is given with with a pair of numbers, where*

the first one indicates the capacity c_{ij} and the second indicates the flow x_{ij}. In Figure 2.12 the value of the flow is 3.

In Figure 2.13 we have an augmenting path, which satisfies $x_{ij} < c_{ij}$ for each forward arc and $x_{ji} > 0$ for each backward arc.

If we increase the flow in Figure 2.12 by the maximal amount in the augmenting path, we obtain an augmented flow, like the one in Figure 2.14. The maximal amount that can be added with respect to the augmented path in Figure 2.13 is one unit, because of the constraints in the arcs $(1,3)$ and $(4,3)$.

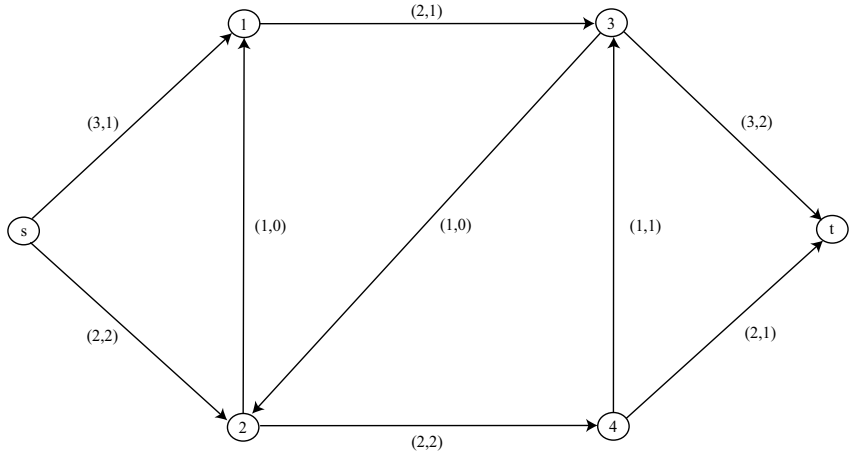

Figure 2.12: A feasible flow.

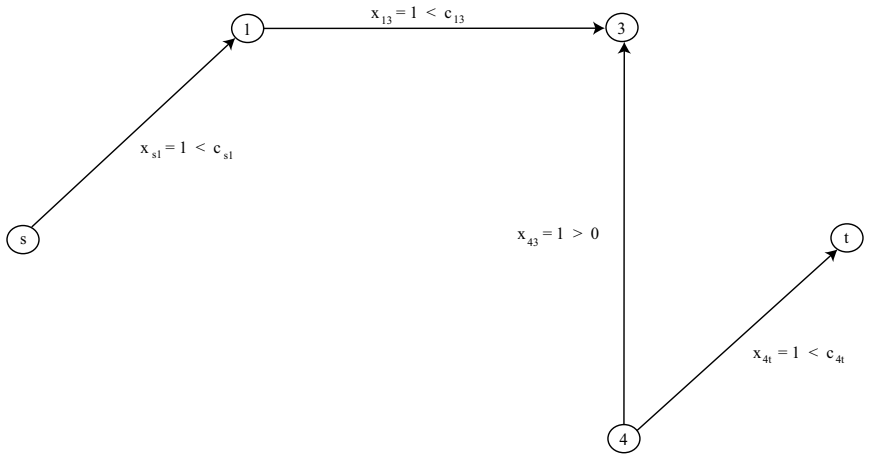

Figure 2.13: An augmenting path.

A *cutset* $cut(S,T)$ is a partition of the vertices in a set S and a set T, where $s \in S$ and $t \in T$. We define it as

$$cut(S,T) = \sum_{i \in S} \sum_{j \in T} c_{ij}, \tag{2.3}$$

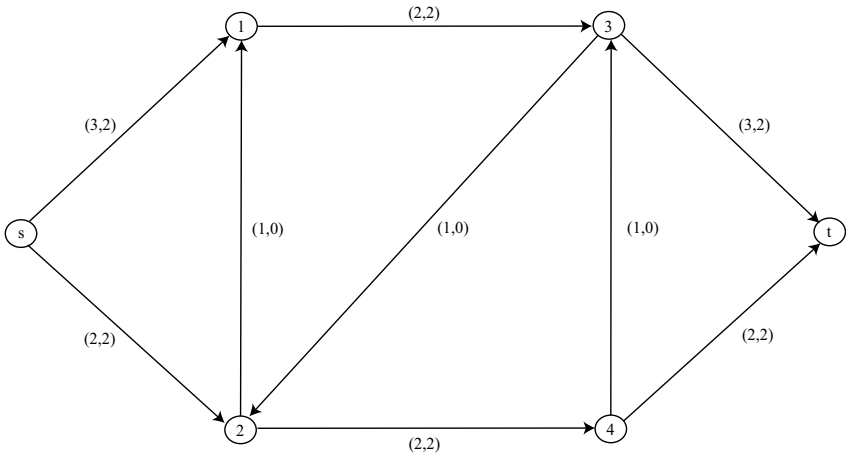

Figure 2.14: An augmented flow.

i.e., the sum of the capacities of all arcs that are directed from S to T.

No flow can exceed any cutset $cut(S,T)$. If we sum the constraints of (2.2), we obtain

$$\begin{aligned} v &= \sum_{i\in S}\Big(\sum_{j} x_{ij} - \sum_{j} x_{ji}\Big) \\ &= \sum_{i\in S}\sum_{j\in S}(x_{ij} - x_{ji}) + \sum_{i\in S}\sum_{j\in T}(x_{ij} - x_{ji}) \\ &= \sum_{i\in S}\sum_{j\in T}(x_{ij} - x_{ji}) \end{aligned} \tag{2.4}$$

We now have the equality between the value v of any flow and any cutset $cut(S,T)$. Then we can say since $x_{ji} \geq 0$ and $x_{ij} \leq c_{ij}$

$$v \leq \sum_{i\in S}\sum_{j\in T} c_{ij} = cut(S,T). \tag{2.5}$$

Theorem 2.4 (Augmenting path Theorem)**.** *A flow is maximal if and only if it admits no augmenting path from s to t.*

Proof. The proof is given by Lawler [14]. □

Theorem 2.5 (Integral flow Theorem)**.** *If all arc capacities are integers there is a maximal flow which is integral.*

Proof. The proof is given by Lawler [14]. □

Chapter 3

Introduction to preemptive open shops

In this chapter we list related works about open shops and several different methods for the resolution of the makespan problem for open shops. We also introduce some new concepts for the preemptive open shop.

3.1 Method for the resolution of the makespan problems with fixed number of processors

We will consider preemptive open shops with p multiprocessors. If we have m processors in the set $\mathcal{P}$ then these multiprocessors are given as $G_1 = \{P_1, \ldots, P_{s_1}\}$, $G_2 = \{P_{s_1+1}, \ldots, P_{s_2}\}$, $\ldots, G_p = \{P_{s_{p-1}}, \ldots, P_n\}$. Where we have $1 < s_1 < s_2 < \ldots < s_{p-1} < m$ and each $s_i \in \mathbb{N}$, $\forall i = 1, \ldots, p-1$.

The resolution follows Brucker [7]. First, we define the concept of a *configuration*. A configuration K is a vector of zeros and ones. The vector is of dimension p, equal to the number of multiprocessors. Thus, it is possible to attribute to each multiprocessor a one or a zero. If we write $K = (K_1, K_2, \ldots, K_p)$, then K_l is the value for the multiprocessor G_l. If $K_l = 0$, then there is no group operation on multiprocessor G_l. If $K_l = 1$, then the multiprocessor is performing a group operation.

If we have a schedule, then we extend the notation and introduce $\mathcal{K}$. $\mathcal{K}$ is a matrix of dimension $|K| \times p$, where $|K|$ is denote as the number of all configurations. Remark that $|K|$ is integral but it does not mean the intervals are integral. The columns of $\mathcal{K}$ are the configurations for the appropriate time intervals in the schedule, meaning K^s is the column in $\mathcal{K}$ corresponding to the s-th interval in the schedule.

Example 3.1. *We take again the Example 2.1 and give the corresponding configurations for this schedule. For the first interval we have* $K^1 = (0, 0)$ *be-*

p=0	1
p=1	2
p=2	4
p=3	8
p=4	16
p=k	2^k

Table 3.1: The number of different configurations for a given number p of multiprocessors

cause no multiprocessor is executing a group operation. Furthermore, we have $K^2 = (1,0), K^3 = (0,1), K^4 = (1,0)$ *and* $K^5 = (0,0)$. *We can also give the configurations directly in the matrix*

$$\mathcal{K} = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Following Brucker [7], we search all possible configurations independent on their length. If we consider first the trivial case without multiprocessors, then we have $\mathcal{K} = (\ 0 \quad 0 \quad 0 \quad \ldots \quad 0\ )$, where the dimension of all column vectors is 1×1, see Table 3.1.

If we consider the case with four processors and two multiprocessors, where $G_1 = \{P_1, P_2\}$ and $G_2 = \{P_3, P_4\}$, then we have only 4 possible configurations.

$$\mathcal{K} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

But if we have two multiprocessors and more then four processors, the number of configurations does not change, because K is only dependent on the number of multiprocessors. In Table 3.1 we present the number of possible configurations for a given p.

If we fix only the number of processors, we have of course a given upper bound for the number of multiprocessors. We can say if the number of processors is fixed, then the number of multiprocessors is also fixed. Consider the following Example in Table 3.2, where $m \leq 3$. Here we have at most one multiprocessor, because a multiprocessor requires more than one processor for a group operation.

Thus for a fixed m we have a fixed number of configurations, so we can enumerate all of them. If we know all configurations, we can compute the makespan

with a corresponding linear problem which is solvable in polynomial time (see H.W. Lenstra, JR. [1983]). We will see such a linear program later for the case of two and three multiprocessors.

number of processors	p maximal
$0 < m \leq 3$	1
$3 < m \leq 5$	2
$5 < m \leq 7$	3
m fixed	$\lfloor \frac{m}{2} \rfloor$

Table 3.2: The maximal number p of multiprocessors for m processors.

3.2 Preemptive open shops without multiprocessors

In this section we consider the model described in the Subsection 2.3.2. There we have n jobs $J_1, \ldots, J_n$, m processors $P_1, \ldots, P_m$ and the corresponding processing times b_{jh} are saved in the matrix B.

	P_1	P_2	P_3	P_4
J_1	1	2	0	0
J_2	0	2	0	1
J_3	0	0	2	1

Figure 3.1: An Example of an open shop without multiprocessors.

3.2.1 Combinatorial resolution

To calculate the makespan, we can also use an edge coloring approach. Therefore, we take a bipartite graph $G = (\mathcal{J}, \mathcal{P}, E)$, where $J_j \in \mathcal{J}$ is the vertex corresponding to the j-th job for all $j = 1, \ldots, n$ and $P_h \in \mathcal{P}$ is the vertex corresponding to the h-th processor for all $h = 1, \ldots, m$. Furthermore, we construct b_{jh} edges between J_j and P_h.

A t-coloring of the edges of G gives us a schedule with makespan t. An edge $\{J_j, P_h\}$ gets color k if and only if the job J_j is performed on processor P_h at the interval k. Due to the König's Theorem 2.2 we can show

$$\Delta(G) = \max\{\max_h \sum_j b_{jh}, \max_j \sum_h b_{jh}\}$$

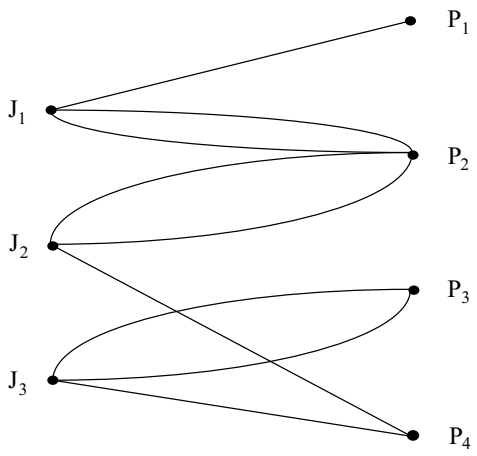

Figure 3.2: The corresponding bipartite multigraph G to the open shop given in Figure 3.1.

and

$$\Delta(G) = C_{max}.$$

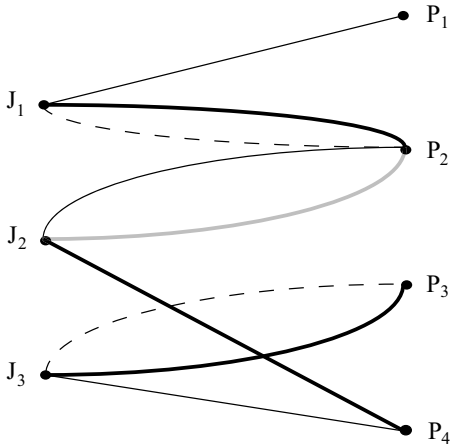

Figure 3.3: The corresponding proper edge coloring to the graph G in Figure 3.2.

3.2.2 Method of Gonzales and Sahni

Gonzales and Sahni (see [12]) have an algorithm to resolve the preemption open shop for any number of processors in polynomial time ($O|pmtn|C_{max}$). Here we will give the main idea of Gonzales and Sahni's algorithm for two processors without preemptions ($O2|\ |C_{max}$). The algorithm gives us a solution in linear time, the makespan of this solution is equal to the makespan of the solution for the preemptive case. For more information see [Gonzales & Sahni, 1976].

We will represent each job by a pair of positive integers. The first integer gives us the processing time for processor P_1 and the second, the processing time for P_2. Now, we partition all jobs in two sets. The set A takes any job J_j with a bigger processing time on P_1 then on P_2, i.e. $b_{j1} \geq b_{j2}$. All other jobs should have a longer processing time on P_2 ($b_{j2} > b_{j1}$) and will be put in the other set B. We consider only the case where these sets are not empty, the other cases are similar to do, see [12].

A is ordered as a sequence $A_1, A_2, \ldots, A_r$, where $A_1 \leq A_2 \leq \ldots \leq A_r$. There, A_1 represents the shortest job in A, and A_r is the longest job. For B we have the ordered sequence $B_1 \leq B_2 \leq \ldots \leq B_s$. The processing time on processor P_h for A_i is denoted by $A_i(h)$, and in the same way we define $B_i(h)$ as the processing time of B_i on processor P_h. So we know, that $A_i(1) \geq A_i(2)$ and $B_i(1) < B_i(2)$.

Now we take the sequence $A_1, A_2, \ldots, A_r$ of jobs on processor P_1; the jobs are performed consecutively, without any preemptions. On processor P_2 we have the same sequence, but with a little delay, because each job can be executed only on one processor at the same time. Hence, on P_2 we start with A_1 as soon as possible, and we only have an idle time between tasks on P_2 (see the second construction in Figure 3.4).

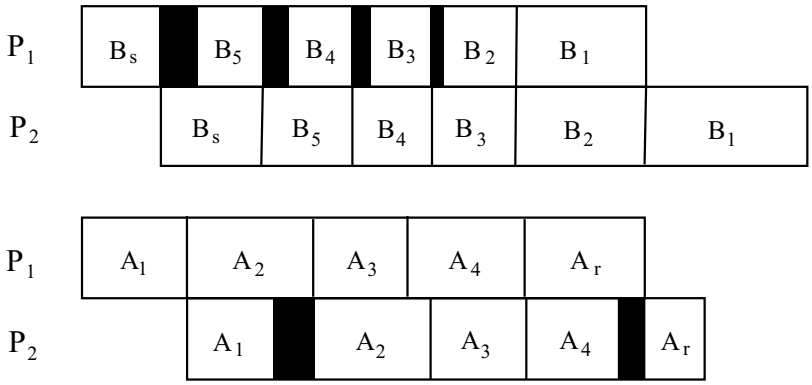

Figure 3.4: Part 1: The schedule construction with the algorithm of Gonzales and Sahni.

Every task A_i on P_2 may start earliest at the time point where A_i is finished on P_1, so we obtain small preemptions in the sequence on P_2. If we push all tasks back, means we start a bit later, but we have no preemptions (see Figure 3.5).

We do the same with all tasks of B, the only difference is that we start the sequence with B_s and not with B_1. We then have the preemptions on P_1 and push forward the processing tasks, so the sequence ends earlier and without preemptions.

Then we put the two parts of A and B together so that we have no idle time on processor P_1, on P_2 or both. In Figure 3.6 we have the case without any idle time on processor P_1, and in the second case with idle time on the P_2.

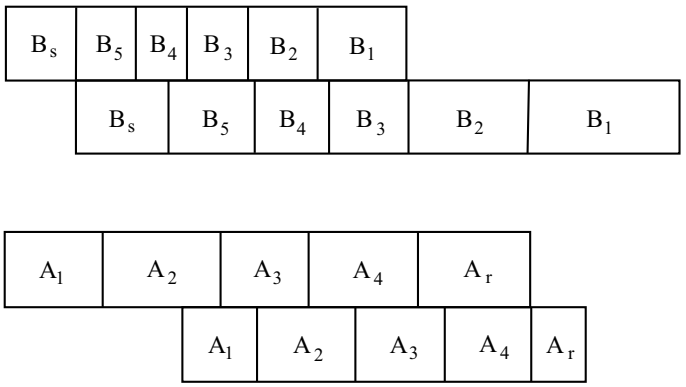

Figure 3.5: Part 2: The transformation of the schedule so as to avoid preemptions between the processes.

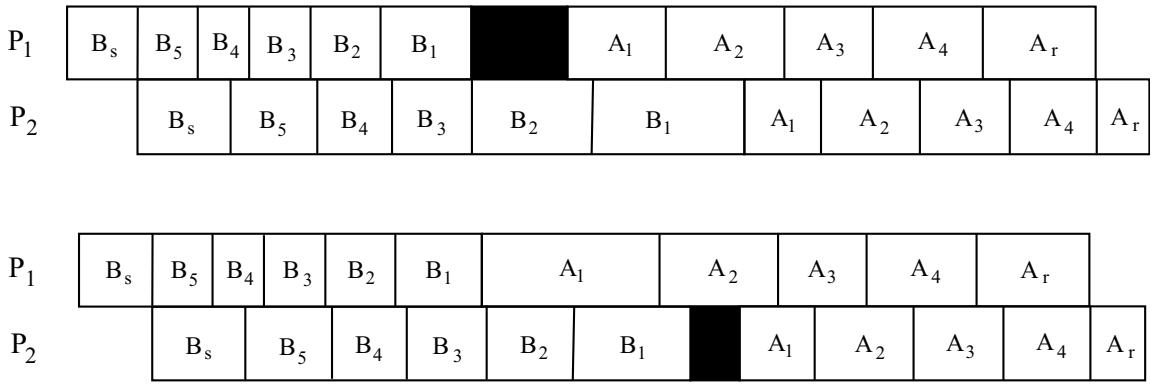

Figure 3.6: Part 3: Putting together set A and B.

Finally we have two possible configurations: first one has the idle time on P_1; in this case we take B_s at the end of all processes of P_1 and push forward all other tasks until the idle has disappeared or B_s will be executed on P_1. The second possibility is where A_r has the idle time on P_2. In Figure 3.7 we take the Example where we have $A_r < B_s$. AS a result we replace A_r at the beginning of the schedule.

Theorem 3.1. *For open shop with two multiprocessors the method of Gonzales and Sahni delivers an optimal schedule. The makespan of this schedule obtains the makespan of an optimal preemptive schedule.*

For the case with more then two multiprocessor see [12].

3.2.3 Consecutive edge coloring for an open shop with two processors

The idea of Gonzales and Sahni, in the case of two processors without preemptions, can be translated into graph theory. Therefore we construct in the same

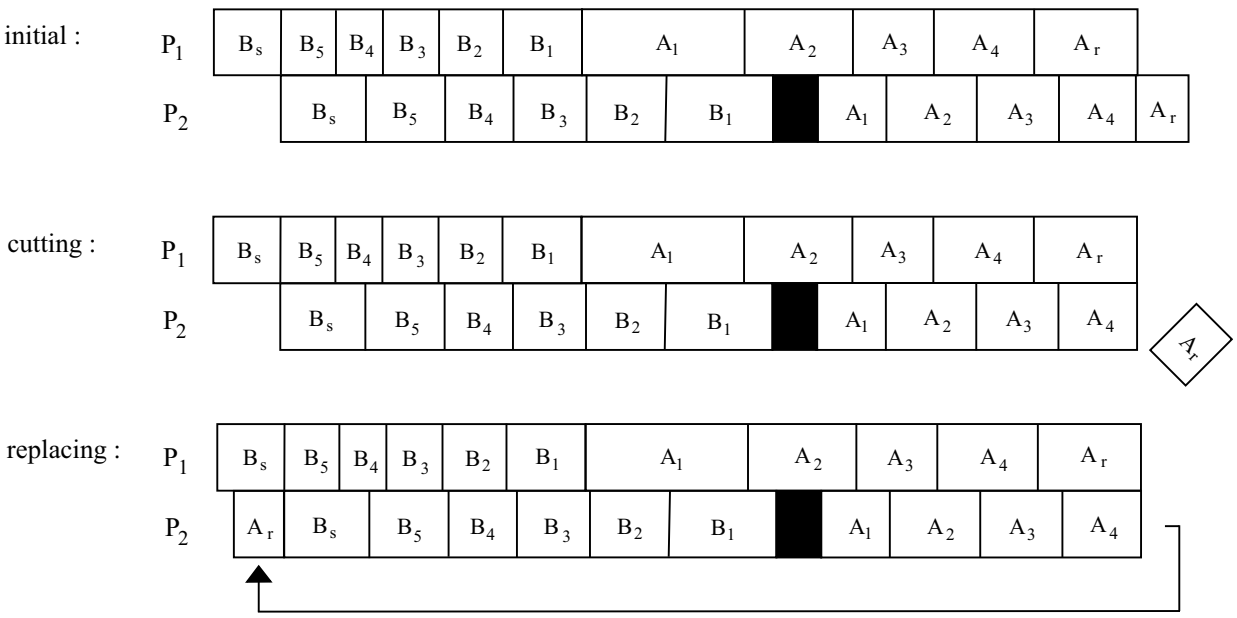

Figure 3.7: Part 4: Optimizing the schedule by cutting the end A_r and attaching it at the start.

way a bipartite multigraph $G = (\mathcal{J}, \mathcal{P}, E)$ as in the Subsection 3.2.1.

But instead of using edge coloring, we use an interval edge coloring. The result of this interval coloring is that we obtain a schedule without preemptions, because of the consecutive colors. Because each color k assigns an interval in the schedule, all edges joining the same vertex describe one entire job on a processor. Therefore we obtain a schedule with the job performed without a preemption, because the consecutive colors represent consecutive intervals.

Remark 3.1. *If G is a bipartite multigraph, then such an interval coloring can be interpreted as a timetable in which the lessons for each class and each teacher are without interruptions. (Asratian et al. in [2].)*

3.3 Preemptive open shop with two multiprocessors

In this case we divide the set $\mathcal{P}$ into two groups, G_1 and G_2, of processors where $G_1 = \{P_1, \ldots, P_s\}$ and $G_2 = \{P_{s+1}, \ldots, P_m\}$. In addition to the individual operations O_{jh} involving only one processor P_h, there are group operations $\hat{O}_{j1}$ and $\hat{O}_{j2}$, that describe the performing of job J_j on multiprocessor G_1 and G_2 respectively. The processing time of job J_j on multiprocessor G_l is given by a_{jl}.

3.3.1 The fractional model

We have to respect the following assumptions for the preemptive open shop with fractional preemptions:

(A1) b_{jh} is an integer number $\forall j = 1, \ldots, n$ and $\forall h = 1, \ldots, m$;

(A2) a_{jl} is an integer number $\forall j = 1, \ldots, n$ and $\forall l = 1, \ldots, p$;

(A3) each processor performs only one operation at a time;

(A4) each multiprocessor performs only one operation at a time;

(A5) two operations (either individual or group) of the same job are never done at the same time, i.e. in parallel;

(A6F) the processing of any operation on any processor (or multiprocessor) may be interrupted at any time and could be resumed later at any time. In the meantime, the processor (or multiprocessor) could be used by any other operation;

3.3.2 The integral model

Considering the assumptions in the preceding fractional model, the integral model takes the same conditions, except only for (A6F) will be replaced by:

(A6I) the processing of any operation on any processor (or multiprocessor) can be interrupted only at integer time points and could resume later at any integer time points. In the meantime, the processor (or multiprocessors) could be used by any other operation;

3.3.3 A normal form of the schedule

We will introduce some necessary definitions to obtain a normal form for all feasible schedules for the open shop with two multiprocessors.

1. $d(P_h) = \sum_{j=1}^{n} b_{jh}$: the total processing time of all individual operations on processor P_h;
2. $d(J_j) = \sum_{h=1}^{m} b_{jh}$: the total processing time of all individual operations of job J_j;
3. $\Delta(G_l) = \sum_{j=1}^{n} a_{jl}$: the total processing time of all group operations done on multiprocessor G_l;

type of operations	interval	time interval
$(g,1),(i,2)$	(a)	$[0, \Delta(G_1) - r]$
$(g,1),(g,2)$	(b)	$[\Delta(G_1) - r, \Delta(G_1)]$
$(i,1),(g,2)$	(c)	$[\Delta(G_1), \Delta(G_1) + \Delta(G_2) - r]$
$(i,1),(i,2)$	(d)	$[\Delta(G_1) + \Delta(G_2) - r, t]$

Table 3.3: The normal form.

If a feasible schedule with makespan t exists, then we can transform the schedule into a new schedule where all group operations on G_1 are continuously performed during $\Delta(G_1)$ units, and all group operations on G_2 are performed continuously during $\Delta(G_2)$ units. The overlap r is the time where both group operations on G_1 and G_2 are processed simultaneously.

We can see that

$$r_{min} = \max\{0, \Delta(G_1) + \Delta(G_2) - t\}$$

is the lower bound and

$$r_{max} = \min\{\Delta(G_1), \Delta(G_2)\}$$

describes the upper bound of the overlap r.

In the special case of two multiprocessors we distinguish four different types of operations:

1. an individual operation O_{jh} is of type (i,l), if it has to be performed on processor P_h, where $P_h \in G_l$ $(l = 1, 2)$;
2. a group operation $\hat{O}_{jl}$ is of type (g,l), if it has to be performed on multiprocessor G_l $(l = 1, 2)$;

We can rearrange the schedule so that the time is divided in four intervals that are dependent on the type of operations. The four intervals are listed in the following Table 3.3 and a possible form of the schedule is given in Figure 3.8.

Remark 3.2. *It is always possible that one of the intervals $(a), (b), (c)$ or (d) is missing, i.e. its length is equal 0.*

3.3.4 Linear programming formulation for (PF)

LP for the (PF)

In this part we reproduce the linear program shown in de Werra, Kis and Kubiak [18]. But first we need the following decision variables:

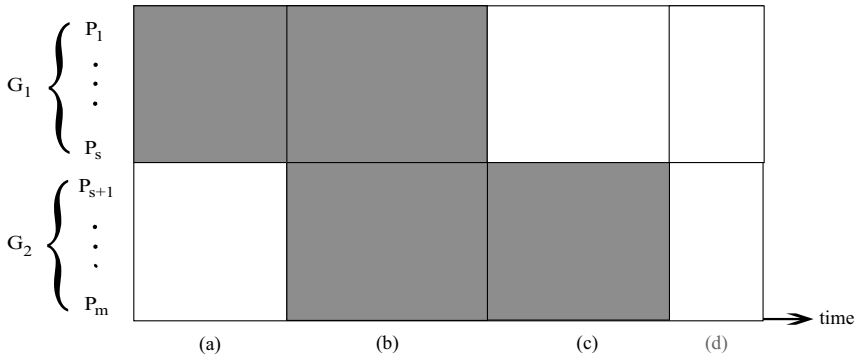

Figure 3.8: The normal form of a schedule for two multiprocessors.

- r : the length of part (b);
- w : the length of part (d);
- x_{jl} : the amount of time group operation $\hat{O}_{jl}$ is processed in part (b);
- y_{jh} : the amount of time individual operation O_{jh} is processed in part (d);

The objective is to minimize the makespan of the schedule. The schedule takes $(w + \Delta(G_1) + \Delta(G_2) - r)$ time units, so we will have to minimize the difference

$$w - r.$$

Furthermore we have following constraints. In the interval (d) the individual processing time does not exceed b_{jh},

$$0 \leq y_{jh} \leq b_{jh},\ \forall j = 1, \ldots, n,\ \forall h = 1, \ldots, m.$$

The processing of $\hat{O}_{jl}$ on multiprocessor G_l takes maximal a_{jl} time units in interval (b), so

$$0 \leq x_{jl} \leq a_{jl},\ \forall j = 1, \ldots, n,\ \forall l = 1, 2.$$

In interval (a) we can not exceed $(\Delta(G_1) - r)$ time units for all individual and group operations of job J_j,

$$\sum_{h=s+1}^{m} (b_{jh} - y_{jh}) + a_{j1} - x_{j1} \leq \Delta(G_1) - r,\ \forall j = 1, \ldots, n$$

In the interval (a), no processor performs longer then $(\Delta(G_1) - r)$ time units,

$$\sum_{j=1}^{n} b_{jh} - \sum_{j=1}^{n} y_{jh} \leq \Delta(G_1) - r,\ \forall h = s + 1, \ldots, n.$$

In (b) the multiprocessor G_l has to execute group operations for exact r time units,

$$\sum_{j=1}^{n} x_{jl} = r,\ l = 1, 2.$$

The total group processing time of J_j takes exactly $a_{j1} + a_{j2}$ time units. But in the interval (b) it does not exceed r time units,

$$x_{j1} + x_{j2} \leq r,\ \forall j = 1, \ldots, n.$$

In interval (c) we can not exceed $(\Delta(G_2) - r)$ time units for all individual and group operations of job J_j,

$$\sum_{h=1}^{s} (b_{jh} - y_{jh}) + a_{j2} - x_{j2} \leq \Delta(G_2) - r,\ \forall j = 1, \ldots, n.$$

In interval (c), no processor performs longer then $(\Delta(G_2) - r)$ time units,

$$\sum_{j=1}^{n} b_{jh} - \sum_{j=1}^{n} y_{jh} \leq \Delta(G_2) - r,\ \forall h = 1, \ldots, s.$$

The total individual processing time of job J_j takes exactly $d(J_j)$ time units. It can not exceed w time units in the interval (d),

$$\sum_{h=1}^{m} y_{jh} \leq w,\ \forall j = 1, \ldots, n.$$

In (d) the individual processing time can not exceed w for any processor P_h,

$$\sum_{j=1}^{n} y_{jh} \leq w,\ \forall h = 1, \ldots, m.$$

The linear program could then be formulated as follows,

$$\begin{aligned}
\min(w-r) \quad & \text{s.t.} && && (3.1)\\
\sum_j b_{jh} - (\Delta(G_2) - r) \le \sum_j y_{jh} \quad & \le && w,\ \forall h \in G_1 \\
\sum_j b_{jh} - (\Delta(G_1) - r) \le \sum_j y_{jh} \quad & \le && w,\ \forall h \in G_2 \\
\sum_h y_{jh} \quad & \le && w,\ \forall j \\
0 \le y_{jh} \quad & \le && b_{jh},\ \forall j,h \\
\sum_j x_{j1} \quad & = && r \\
\sum_j x_{j2} \quad & = && r \\
x_{j1} + x_{j2} \quad & \le && r,\ \forall j && (3.2)\\
0 \le x_{jl} \quad & \le && a_{jl},\ \forall j,l \\
\sum_{h\in G_1} (b_{jh} - y_{jh}) + a_{j2} - x_{j2} \quad & \le && \Delta(G_2) - r,\ \forall j \\
\sum_{h\in G_2} (b_{jh} - y_{jh}) + a_{j1} - x_{j1} \quad & \le && \Delta(G_1) - r,\ \forall j
\end{aligned}$$

This fractional model can be solved in strongly polynomial time. Strongly polynomial means its running time is polynomial in the dimension of the parameters done in this Example as n, m and p. A proof is shown in de Werra, Kis and Kubiak [18].

Schedule construction

Once the linear program is solved, we have x_{jl} and y_{jh}. These values give us the processing time corresponding to the intervals (a) and (c). If (a) and (c) are determined we continue with (b) and (d). We can present each interval as matrix $T_{(a)}, T_{(b)}, T_{(c)}$ and $T_{(d)}$:

Interval (a) **:**

$$T_{(a)} = \begin{array}{c|cccccc} & \mathrm{J}_1 & \cdots & \mathrm{J}_j & \cdots & \mathrm{J}_n & \\ G_1 & x_{11} & \cdots & x_{j1} & \cdots & x_{n1} & = \Delta(G_1) - r \\ P_{s+1} & & & & & & \\ \vdots & & & \vdots & & & \\ P_h & \cdots & & y_{jh} & \cdots & & \le \Delta(G_1) - r \\ \vdots & & & \vdots & & & \\ P_m & & & & & & \\ & \multicolumn{5}{c}{\le \Delta(G_1) - r} & \end{array}$$

Interval (b) **:**

$$T_{(b)} = \begin{array}{c|ccccc|l} & \mathrm{J}_1 & \cdots & \mathrm{J}_j & \cdots & \mathrm{J}_n & \\ G_1 & a_{11}-x_{11} & \cdots & a_{j1}-x_{j1} & \cdots & a_{n1}-x_{n1} & = r \\ G_2 & a_{12}-x_{12} & \cdots & a_{j2}-x_{j2} & \cdots & a_{n2}-x_{n2} & = r \\ & \multicolumn{5}{c}{\leq r} & \end{array}$$

Interval (c) **:**

$$T_{(c)} = \begin{array}{c|ccccc|l} & \mathrm{J}_1 & \cdots & \mathrm{J}_j & \cdots & \mathrm{J}_n & \\ P_1 & & & & & & \\ \vdots & & & \vdots & & & \\ P_h & \cdots & & y_{jh} & \cdots & & \leq \Delta(G_2)-r \\ \vdots & & & \vdots & & & \\ P_s & & & & & & \\ G_2 & x_{12} & \cdots & x_{j2} & \cdots & x_{n2} & = \Delta(G_2)-r \\ & \multicolumn{5}{c}{\leq \Delta(G_2)-r} & \end{array}$$

Interval (d) **:**

$$T_{(d)} = \begin{array}{c|ccccc|l} & \mathrm{J}_1 & \cdots & \mathrm{J}_j & \cdots & \mathrm{J}_n & \\ P_1 & & & & & & \\ \vdots & & & \vdots & & & \\ P_h & \cdots & & b_{jh}-y_{jh} & \cdots & & \leq w \\ \vdots & & & \vdots & & & \\ P_s & & & & & & \\ P_{s+1} & & & & & & \\ \vdots & & & \vdots & & & \\ P_{h'} & \cdots & & b_{jh'}-y_{jh'} & \cdots & & \leq w \\ \vdots & & & \vdots & & & \\ P_m & & & & & & \\ & \multicolumn{5}{c}{\leq w} & \end{array}$$

We now give the idea for the construction of the schedule in a example, where we have integers x_{jl} and y_{jh} (see Cereghetti [8]). For the case with any x_{jl} and y_{jh} see Gonzales and Sahni in [12]. We will present only the case for the first interval (a), as the other intervals are similar.

Consider the matrix, where $G_1 = \{P_1, P_2\}$,

$$T_{(a)} = \begin{array}{c|ccc} & J_1 & J_2 & J_3 \\ G_1 & 3 & & \\ P_3 & 1 & 2 & \\ P_4 & & 2 & 2 \\ P_5 & & 1 & 1 \end{array}$$

Then the graph corresponding to the matrix is given in Figure 3.9. The graph contains a 5-edge coloring. Each color represents a time interval in the schedule.

Finally, we obtain the schedule in Figure 3.10 as the corresponding schedule to the edge coloring in Figure 3.9.

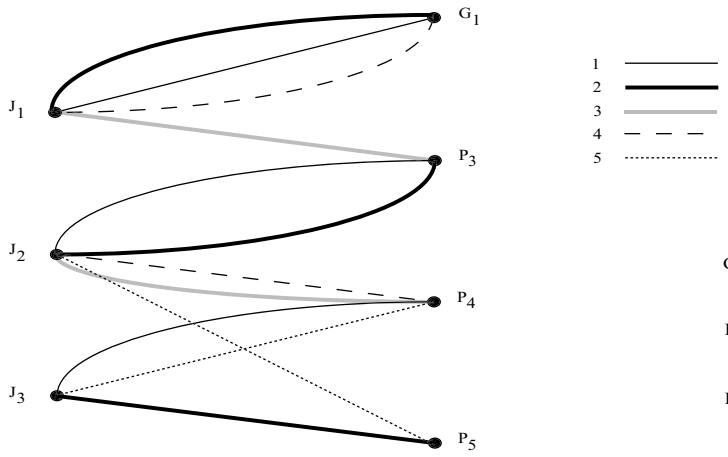

Figure 3.9: The corresponding graph to $T_{(a)}$. The different edge styles assign different colors.

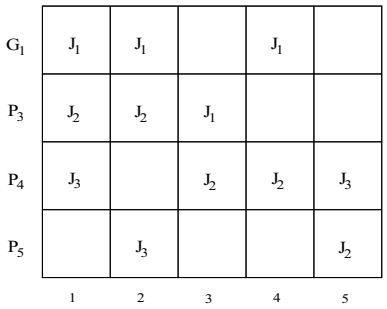

Figure 3.10: The schedule to the edge coloring in Figure 3.9.

3.3.5 The integral problem (PI)

For the integral problem the preemptions can only occur at integral time points. As previously mentioned, we consider only integral data. We can show that fractional makespan can be shorter then integral. The Example is shown in Figure 3.11. It is taken from de Werra, Kis and Kubiak [18].

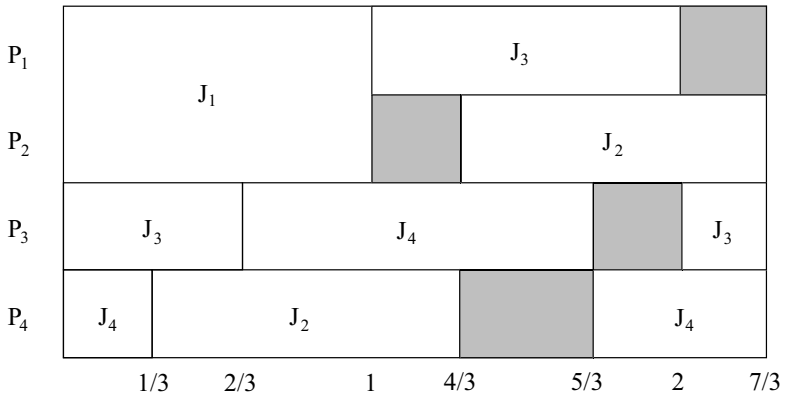

Figure 3.11: A fractional optimal schedule.

They present sufficient conditions which allow us to solve the integral problem in polynomial time. The main idea is based on binary jobs.

Now lets define a *binary job*. A job is a binary job, if its operations are of at most two different types. Thus a binary can not have, for example, a group operation and individual operations on processors in each of the processor groups.

We partition all binary jobs in four sets, $\mathcal{J}^{gi} \cup \mathcal{J}^{ig} \cup \mathcal{J}^{gg} \cup \mathcal{J}^{ii}$ where $\mathcal{J}^{gi}$ consists of all jobs with operations of type $(g,1)$, and either $(i,1)$ or $(i,2)$, but not both. $\mathcal{J}^{ig}$ contains all jobs with operations of type $(g,2)$, and either $(i,1)$ or $(i,2)$, but not both. Furthermore, all jobs in $\mathcal{J}^{gg}$ have only group operations, and all jobs in $\mathcal{J}^{ii}$ have only individual operations.

As we have seen in the Subsection 3.3.4, the minimum makespan is given by $(\Delta(G_1)+\Delta(G_2)+w-r)$. But now we want to show that for the integral model, the minimum is given by $(\Delta(G_1)+\Delta(G_2)+\lceil w^*-r^*\rceil)$. w^* and r^* are taken from any optimal solution to the LP of the fractional problem. So we obtain the following LP for binary jobs:

$$\min(w-r) \quad s.t. \tag{3.3}$$

$$\begin{aligned}
\sum_{j\in\mathcal{J}\setminus\mathcal{J}^{gg}} b_{jh} - (\Delta(G_2)-r) \le \sum_{j\in\mathcal{J}\setminus\mathcal{J}^{gg}} y_{jh} &\le w,\ \forall h\in G_1\\
\sum_{j\in\mathcal{J}\setminus\mathcal{J}^{gg}} b_{jh} - (\Delta(G_1)-r) \le \sum_{j\in\mathcal{J}\setminus\mathcal{J}^{gg}} y_{jh} &\le w,\ \forall h\in G_2\\
\sum_{h\in G_1\cup G_2} y_{jh} &\le w,\ \forall j\in\mathcal{J}^{ii}\\
\sum_{h\in G_1} y_{jh} &\le w,\ \forall j\in\mathcal{J}^{gi}\cup\mathcal{J}^{ig}\\
\sum_{h\in G_2} y_{jh} &\le w,\ \forall j\in\mathcal{J}^{gi}\cup\mathcal{J}^{ig}\\
0\le y_{jh} &\le b_{jh},\ \forall j,h\\
\sum_{j\in\mathcal{J}^{gg}\cup\mathcal{J}^{gi}} x_{j1} &= r\\
\sum_{j\in\mathcal{J}^{gg}\cup\mathcal{J}^{ig}} x_{j2} &= r\\
x_{j1}+x_{j2} &\le r,\ \forall j\in\mathcal{J}^{gg}\\
0\le x_{jl} &\le a_{jl},\ \forall j,l\\
\sum_{h\in G_1} b_{jh} + a_{j2} - (\Delta(G_2)-r) &\le \sum_{h\in G_1} y_{jh} + x_{j2},\ \forall j\in\mathcal{J}^{ig}\\
\sum_{h\in G_2} b_{jh} + a_{j1} - (\Delta(G_1)-r) &\le \sum_{h\in G_2} y_{jh} + x_{j1},\ \forall j\in\mathcal{J}^{gi}
\end{aligned}$$

If w and r are fixed, then this LP is equivalent to a capacitated network flow problem, denoted by $N(w, r)$.

The algorithm to find the minimum makespan in polynomial time is given by the following three steps:

1. Solve the LP (3.3) for the integral problem. Let (w^*, r^*, x^*, y^*) be an optimal solution;

2. Modify the LP (3.3)as follows: replace the objective function by $\min r$, and add the constraint
$$w - r = \lceil w^* - r^* \rceil$$
to the existing constraints. Call the resulting system LP';

3. Solve LP', let (w, r, x, y) be an optimal solution. Find a compatible flow in $N(w, r)$ with w and r fixed to the values of this optimal solution. Convert the solution into an open shop schedule;

This algorithm runs in strongly polynomial time, see [18].

To prove the fact that the solution of the minimum makespan with integral preemptions is optimal, it is necessary to show that r, in Step 3 of the algorithm, is integral. Since this implies integrality of w, it follows that all bounds in $N(w, r)$ are integral. And so we will obtain an integral compatible flow in the $N(w, r)$ which produces an integral solution.

[18] proves the following theorem:

Theorem 3.2. *If all jobs are binary, r is integral in any optimal solution to LP'.*

Finally, we find the optimal integral solution to the makespan minimization problem.

Furthermore, de Werra, Kis and Kubiak i [18] have shown that their algorithm can be simplified if we take only *simple* jobs. A binary job will be called simple, if either it has no individual operations or it has no group operations or all its individual and group operations are on the same group of processors.

They prove the following lemma:

Lemma 3.1. *Given any instance of the makespan minimization problem in the integral model with 2 multiprocessors and all simple jobs, there exists an optimal schedule S with makespan T^* with all the following properties:*

- *the overlap r of group operations attains the minimum:*
$$r = \max\{0, \Delta(G_1) + \Delta(G_2) - T^*\} = r_{\min}(T^*);$$

- *there are no preemptions for the operations on multiprocessors G_1;*

- *there are at most* $r_{\min}(T^*) - 1$ *preemptions for the group operations on* G_2*;*

The proof is given by the in Figure 3.11. There we can see the optimal makespan is not an integer and so the (PI) must obtain a bigger solution.

Remark 3.3. *Kubiak (see [13]) show that the general (PI) problem with two multiprocessors is solvable in polynomial time.*

3.3.6 Hypergraph representation for (PI)

If we consider only integral preemptions, then we have the integral problem (PI). In this case we do not have to use the resolution through the linear programming. Instead, we can choose a combinatorial resolution considering hypergraph edge coloring.

To minimize the makespan, it is sufficient to find a proper edge coloring to the corresponding hypergraph H.

The hypergraph will be constructed by the following steps:

- for each job J_j we construct a vertex J_j;
- for each processor P_h we construct a vertex P_h;
- for each individual operation O_{jh} we construct an edge $\{J_j, P_h\}$
- for each group operation $\hat{O}_{jl}$ we construct a hyperedge $\{J_j, G_l\}$. For example if $G_l = \{P_1, P_2, P_3\}$ then we have a hyperedge $\{J_j; P_1, P_2, P_3\}$.

Let us consider the method in an example. We have the initial table with the given processing times, as in Figure 3.12. The first step is to create the appropriate hypergraph H, represented in Figure 3.13. If we have found H then we can identify a proper t-edge coloring for H, shown in Figure 3.14. The edge coloring with t colors represents a schedule in t units, where each color represents a time interval, see 3.15.

Remark 3.4. *The problem with the combinatorial method hypergraph edge coloring is that it is difficult. To determine, if a* l*-uniform hypergraph is* t*-colorable, is NP-hard for* $l = 2,\ k \geq 3$ *and also for* $k = 2,\ l \geq 3$*. For more information about hypergraph coloring see [9].*

3.4 Open shops in school timetables

As already mentioned in the introduction, we will focus primarily on school timetables, the (PI) problem where we have integral preemptions.

	P_1	P_2	P_3	P_4	G_1	G_2
J_1	1	2	0	0	2	0
J_2	0	1	2	1	0	0
J_3	1	0	2	1	0	1
	G_1		G_2			

Figure 3.12: The processing times.

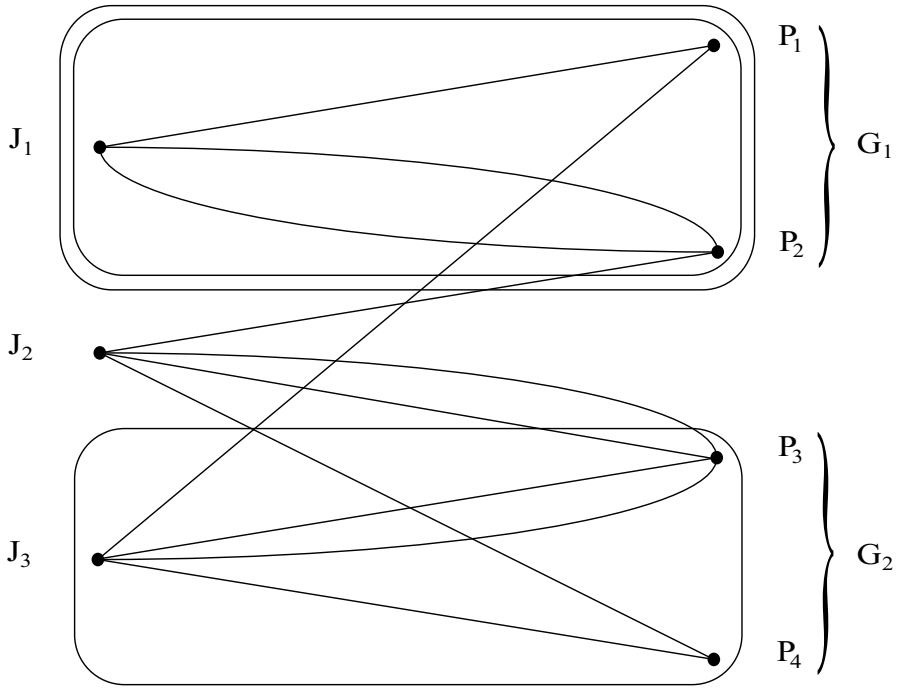

Figure 3.13: The corresponding hypergraph H.

For the integral problem, Asratian and de Werra [1] have shown that the makespan minimization problem is unary $\mathcal{NP}$-hard even for $p = 3$ multiprocessors.

Asratian, Durand and de Werra [17] have also shown that if each job has only group operations or only individual operations - such jobs are called *homogenous*

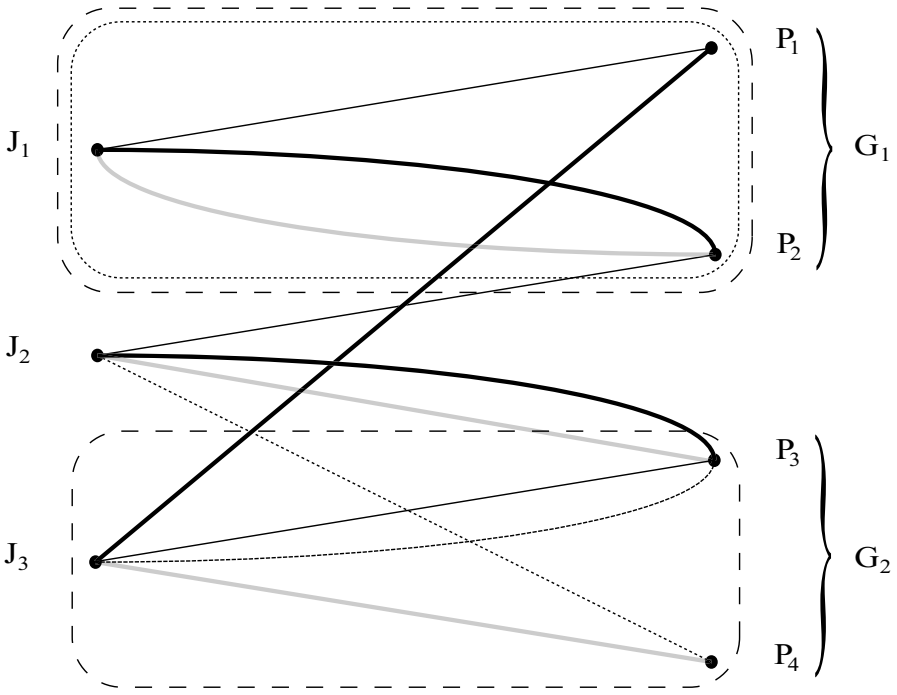

Figure 3.14: A proper edge coloring for H.

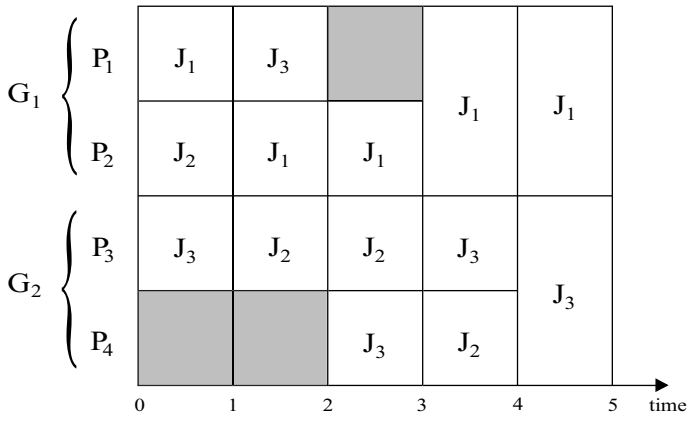

Figure 3.15: The final schedule.

jobs - then the integral model with $p = 3$ is unary $\mathcal{NP}$-hard for the optimization makespan problem.

For these result we introduce the *generalized class-teacher model*. In this model we have m teachers $T_1, \ldots, T_m$ and n classes $C_1, \ldots, C_n$. The classes are partitioned into p disjoint sets $G_1, \ldots, G_p$. There we make a difference between

the individual lectures given by one teacher to one class and the group lectures given by one teacher to one group of classes G_i. The number of lectures are given by b_{jh} and the group lectures are given by a_{jl} where $A = (a_{jl})$ and $B = (b_{jh})$ are the corresponding matrices. Furthermore, we assume all lectures have the same duration, say one period, and the set $\{1, \ldots, t\}$ of all periods is given. The generalized class-teacher model is called a *university timetable*.

The more simple *class-teacher model* could be obtained by taking all $a_{jl} = 0$. Thus we have only individual lectures.

On the other hand we have a special case of class-teacher model if we have two different types of teacher. If we have professors, who give only group lectures and lecturers, who give only lectures, then we call it the *professor-lecturer model*. The professor-lecturer model will be discussed in more detail in Section 5.5.

Additionally, for the university timetable Asratian and de Werra [1] have shown the following theorem:

Theorem 3.3. *The problem of determining, whether there is a university timetable of length t, corresponding to the matrices A and B, is $\mathcal{NP}$-complete even in the case of $t = 3$ and $p \leq 4$.*

The problem is that the existence of a V-sequential coloring is $\mathcal{NP}$-complete for a bipartite multigraph G even if $\Delta(G) = 3$. The $\mathcal{NP}$-completeness was shown by Asratian and Kamalian [1987].

Theorem 3.4. *If it is possible to find a t-sequential coloring of a bipartite graph, then the optimal makespan for the preemptive and the non-preemptive model are equal.*

Proof. The proof is obvious due to König's Theorem for bipartite graphs, see Theorem 2.2. □

We continue with the fractional model of preemptive open shop with multiprocessors in the following Chapter 4. There we start with the model with one multiprocessor and then continue with the case of three. Then we illustrate a possibility to construct the linear programm for $p \geq 4$.

Chapter 4

Preemptive open shops with a fixed number of multiprocessors and fractional preemptions

First we consider the case of one multiprocessor, this case has not been considered in the literature so far, then we continue the case with three multiprocessors. Finally we give an formulation of how to construct the LP for more then three multiprocessors.

4.1 Preemptive open shop with one multiprocessor

4.1.1 Introduction and resolution with known methods

The preemptive open shop with one multiprocessor is a small extension to the model without multiprocessors. We have the same sets of jobs $\mathcal{J}$ and processors $\mathcal{P}$. But we add a multiprocessor, a set of processors, that must perform one job at the same time for all included processors. We denote this group of processors, G, and add to the individual operations one group operation $\hat{O}_j$. The processing time is given by the number of a_j.

It is important to notice that this group operation includes all processors of the set $\mathcal{P}$. Because if G include only the processors $P_1, \ldots, P_s$, $s < m$, then we automatically obtain a second multiprocessor G' that includes all processors $P_{s+1}, \ldots, P_m$. But, for the multiprocessor G' all group operations do not take place, meaning all $a'_j = 0$. This case was discussed in Section 3.3.

If we consider now an open shop with only one multiprocessor, it is not difficult to recognize that in the time when the multiprocessor is processing, no other processes could be done. We logically split the open shop into a multiprocessing part and an individual processing part.

The multiprocessing part is easy to solve, because it does not matter at which moment it takes place; if this part is executing, it is the only one. Considering the fact of the preemptions, this execution can be interrupted and resumed a finite number of times. The total processing time for this multiprocessing part is equal to $\sum a_j$.

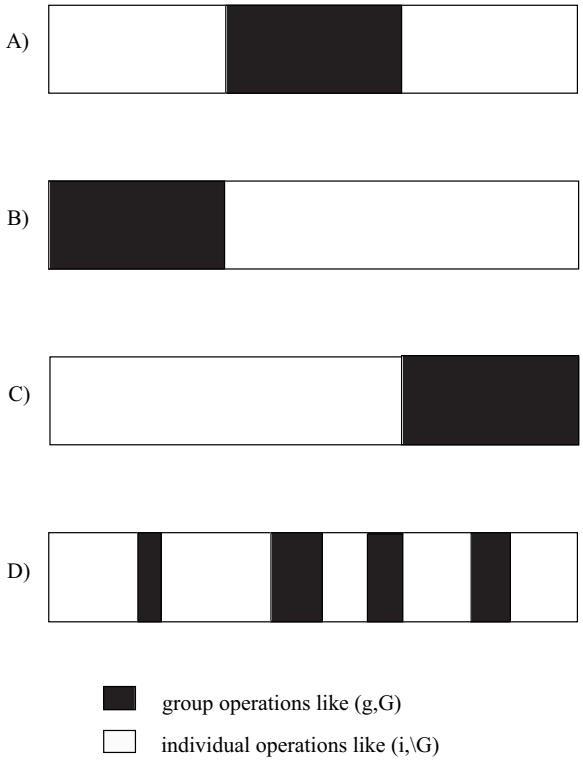

Figure 4.1: Examples of different schedules for preemptive open shops with one multiprocessor.

In Figure 4.1 we can see different examples of schedules. In A) all group operations are between the individual operations, in B) they are at the beginning, in C) at the end. An Example of an arbitrary splitting is shown in D).

After this splitting the individual processing part is reduced to a preemptive open shop without multiprocessors.

Combinatorial method

Here we consider only the case where we have a multiprocessor that does not include all processors, because if it includes them all, we must separate it in two cases, as we have seen in Section 4.1.1. We assume the group operations will be performed first. Then the second part is a proper edge coloring, where we define a bipartite graph $G = (V, W, E)$. V is the set of jobs, W is the set of processors and E is the set of arcs. Each job J_j and processor P_h takes exactly

type of operation	interval	time interval
$(g, G), (i, \backslash G)$	(a)	$[0, \Delta(G)]$
(i, P)	(b)	$[\Delta(G), t]$

Table 4.1: The normal form for one multiprocessor.

one vertex, written as v_j and w_h, but for each processing time unit of the job J_j on the processor P_h, we construct an edge (v_j, w_h).

If we have constructed the corresponding graph, then we use an edge-coloring algorithm, where each color correspond to a time interval of processing. We can take, as an example, the algorithm of Gabow and Kariv in [10].

Let us now consider the case where the multiprocessor is not processing on all processors. We will give the following model with the processors $P_1, \ldots, P_m$, the set of multiprocessor $G = \{P_1, \ldots, P_s\}$ with $s < m$ and the jobs $J_1, \ldots, J_n$.

Now we define a normal form for a schedule, because we will perform the group operation first and after it, the remaining individual operations for the set G. We will give here a special notation for the operation types in the case with only one multiprocessor; we will call (g, G) group operation on multiprocessor G, (i, G) individual operations on any processor included in G and $(i, \backslash G)$ for the individual operations on any processor not included in G.

The normal form of a schedule is shown in Figure 4.2 and the possible types of operations for the intervals (a) and (b) are given in Table 4.1:

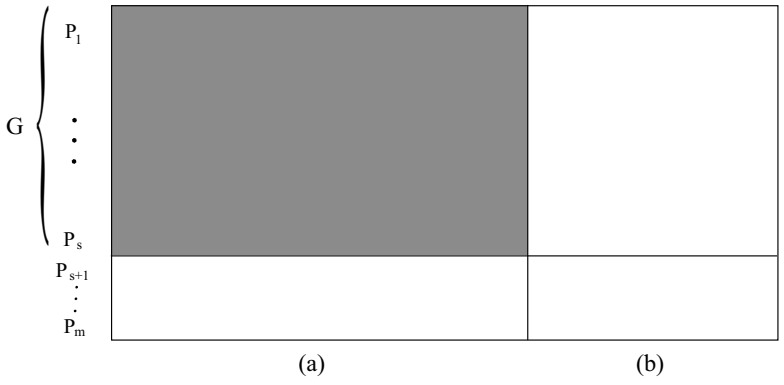

Figure 4.2: The normal form of a schedule with one multiprocessor.

Now we construct a bipartite graph $R = (\mathcal{J}, \mathcal{P}, E)$ in the same way as before. Each job J_j becomes a vertex, $v_j \in \mathcal{J}$, and analog each processor P_h transforms into $w_h \in \mathcal{P}$. Then for each individual operation we give an edge (v_j, w_h). But now we have to introduce the group operations, so we add a vertex G for the

multiprocessor and add an edge (G, v_j), if the job J_j will be performed by the multiprocessor G. You can see an Example of such a constructed graph in Figure 4.3. There we have five processors, the multiprocessor $G = \{P_1, P_2, P_3\}$ and four jobs $J_1, \ldots, J_4$.

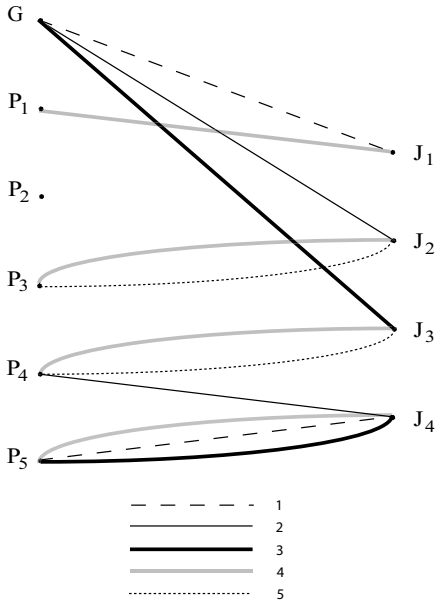

Figure 4.3: An Example of a multigraph R with one multiprocessor and the corresponding edge coloring.

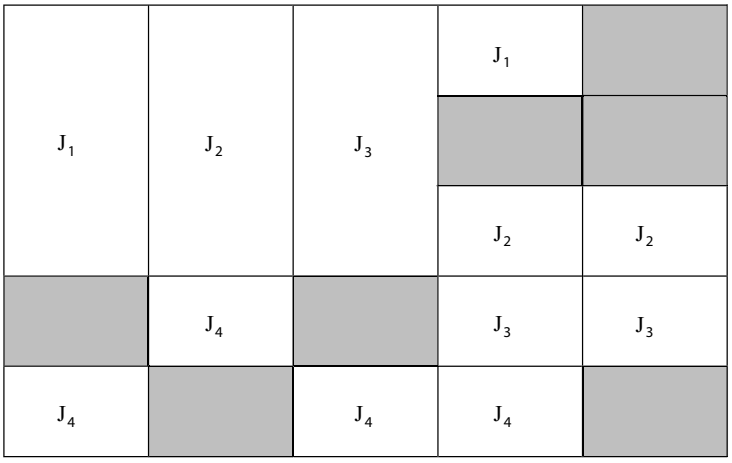

Figure 4.4: The schedule based on the graph in Figure 4.3.

The edge coloring is separated in three parts. The first part is to construct a graph including only the edges of joining in G and to color all the edges by a

consecutive edge coloring with colors form 1 to s. The second part is to add all edges (v_j, w_h) where $P_h \notin G$ to the graph and to continue the edge coloring with the possible colors 1 to s. The edges which can not be colored with the colors 1 to s and the remaining edges, added now, are colored in part three with new colors $s+1$ to t. Thus, we obtain a schedule with makespan t. The final coloration of the graph in Figure 4.3 is given in the schedule in Figure 4.4.

Theorem 4.1. *The obtained schedule for the optimization makespan problem is optimal.*

Proof. During exactly $d_R(G)$ time units we execute the group operations, and we can only perform individual operations O_{jh} with $h > s$ at the same time, because all other processors P_h with $h \leq s$ are in progress of $\hat{O}_j$. Whether the performing of the group operations is in the beginning, the middle, the end or splitted does not matter, the time will always be the same.

For the remaining operations, it is already proven, because we know that the optimal solution for the (PI) is the same as for the (PF), because we have no multiprocessor at this time. □

4.2 Preemptive open shop with three multiprocessors

In this section we introduce the example of the preemptive open shop with three multiprocessors. So we partition the set $\mathcal{P}$ in three groups of processors, we take $G_1 = \{P_1, \ldots, P_{s_1}\}, G_3 = \{P_{s_1+1}, \ldots, P_{s_2}\}$ and $G_3 = \{P_{s_2+1}, \ldots, P_m\}$. We have $1 < s_1 < s_2 < m$.

As in the analogous case of two multiprocessors, we try to find a linear problem with a given form of the schedule (Table 4.2). See also Figure 4.5 for the order of the intervals.

Now we want to construct en LP. For this objective we introduce some necessary variables:

$x_{jl}^{(1)}$: the total group processing time of all $\hat{O}_{jl}$ in the time interval (a_1) or (a_2);

$y_{jh}^{(1)}$: the total individual processing time of all O_{jh} in the time interval (a_1) or (a_2);

$x_{jl}^{(2)}$: the total group processing time of all $\hat{O}_{jl}$ in the time interval (b_1) or (b_2);

$y_{jh}^{(2)}$: the total individual processing time of all O_{jh} in the time interval (b_1) or (b_2);

$x_{jl}^{(3)}$: the total group processing time of all $\hat{O}_{jl}$ in the time interval (c_1) or (c_2);

types of operations	interval
$(g,1),(g,2),(i,3)$	(a_1)
$(i,1),(i,2),(g,3)$	(a_2)
$(g,1),(i,2),(g,3)$	(b_1)
$(i,1),(g,2),(i,3)$	(b_2)
$(i,1),(g,2),(g,3)$	(c_1)
$(g,1),(i,2),(i,3)$	(c_2)
$(g,1),(g,2),(g,3)$	(r)
$(i,1),(i,2),(i,3)$	(w)

Table 4.2: Tho normal form for three multiprocessors.

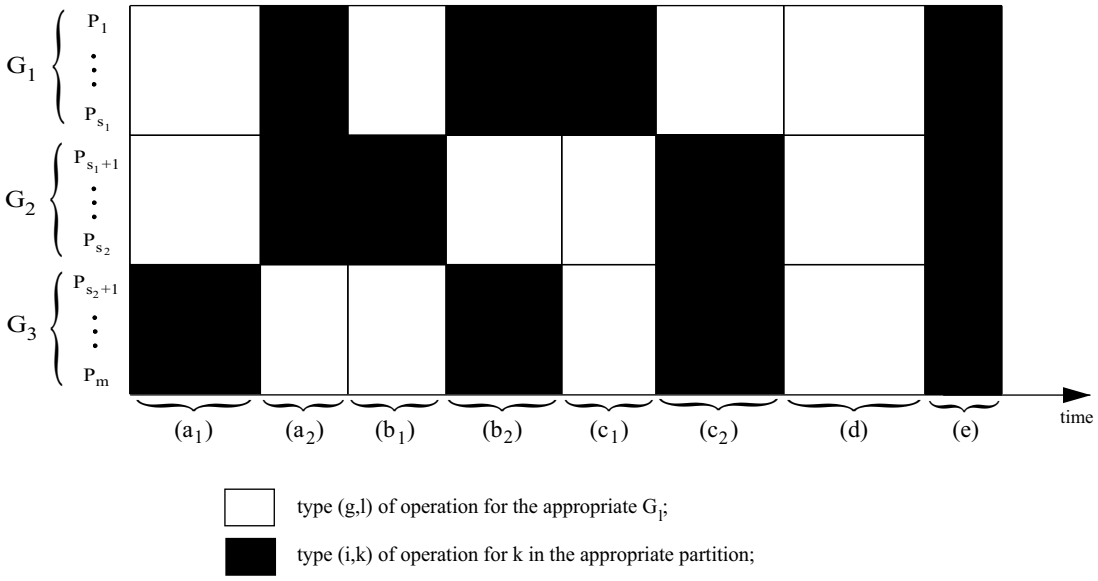

Figure 4.5: The normal form of a schedule with three multiprocessors.

$y_{jh}^{(3)}$: the total individual processing time of all O_{jh} in the time interval (c_1) or (c_2);

a_i : the length of the interval (a_i), $i = 1, 2$;

b_i : the length of the interval (b_i), $i = 1, 2$;

c_i : the length of the interval (c_i), $i = 1, 2$;

r : the length of the interval (r);

w : the length of the interval (w);

Now we can start to formulate the objective and its constraints. The makespan equals $(a_1+a_2+b_1+b_2+c_1+c_2+r+w)$. However, we have the following equations

$$\begin{aligned} a_2 &= \Delta(G_3) - b_1 - c_1 - r \\ b_2 &= \Delta(G_2) - a_1 - c_1 - r \\ c_2 &= \Delta(G_1) - a_1 - b_1 - r \end{aligned}$$

With these equations and the constant $\Delta(G_l)$, we receive the shorter objective

$$\min(w - 2r - a_1 - b_1 - c_1).$$

The constraints are more involved. First, we give the bounds for x and y:

$$0 \leq \sum_{i=1}^{3} y_{jh}^{i} \leq b_{jh},\ \forall j = 1, \ldots, n,\ \forall h = 1, \ldots, m;$$

$$0 \leq \sum_{i=1}^{3} x_{jl}^{i} \leq a_{jl},\ \forall j = 1, \ldots, n,\ l = 1, 2, 3;$$

Then we consider the constraints for each job in interval (a_1). There we have: for the jobs

$$\sum_{h=s_2+1}^{m} y_{jh}^{(1)} + x_{j1}^{(1)} + x_{j2}^{(1)} \leq a_1,\ \forall j = 1, \ldots, n;$$

for the processors with individual operations

$$\sum_{j=1}^{n} y_{jh}^{(1)} \leq a_1,\ \forall h = s_2 + 1, \ldots, m;$$

and for the multiprocessors we have

$$\sum_{j=1}^{n} x_{jl}^{(1)} = a_1,\ l = 1, 2;$$

The next step is to determine the constraints for each job in (a_2). There we have: for the jobs

$$\sum_{h=1}^{s_1} y_{jh}^{(1)} + \sum_{h=s_1+1}^{s_2} y_{jh}^{(1)} + x_{j3}^{(1)} \leq \Delta(G_3) - b_1 - c_1 - r,\ \forall j = 1, \ldots, n;$$

for the processors with individual operations

$$\sum_{j=1}^{n} y_{jh}^{(1)} \leq \Delta(G_3) - b_1 - c_1 - r,\ \forall h = 1, \ldots, s_2;$$

and finally for the multiprocessor G_3 we have the constraint

$$\sum_{j=1}^{n} x_{j3}^{(1)} = \Delta(G_3) - b_1 - c_1 - r;$$

For the other intervals $(b_1), (b_2), (c_1)$ and (c_2) we have similar constraints. We present them in (4.1). For (r), the overlap, we have all three multiprocessors running. So we have: for the jobs

$$\sum_{l=1}^{3} a_{jl} - \sum_{l=1}^{3}\sum_{i=1}^{3} x_{jl}^{(i)} = r, \ \forall j = 1, \ldots, n;$$

and the multiprocessors

$$\sum_{j=1}^{n} a_{jl} - \sum_{j=1}^{n}\sum_{i=1}^{3} x_{jl}^{(i)} = r, \ l = 1, 2, 3;$$

In interval (e), we have: for the jobs

$$d(J_j) - \sum_{h=1}^{m}\sum_{i=1}^{3} y_{jh}^{(i)} \leq w, \ \forall j = 1, \ldots, n;$$

and analog for the processors

$$d(P_h) - \sum_{j=1}^{n}\sum_{i=1}^{3} y_{jh}^{(i)} \leq w, \ \forall h = 1, \ldots, m;$$

With all these constraints we arrive to the following LP:

$$\min(w - 2r - a_1 - b_1 - c_1) \tag{4.1}$$

$$\begin{aligned}
0 \le \sum_{i=1}^{3} y_{jh}^{(i)} &\le b_{jh},\ \forall j = 1, \ldots, n,\ \forall h = 1, \ldots, m \\
0 \le \sum_{i=1}^{3} x_{jl}^{(i)} &\le a_{jl},\ \forall j = 1, \ldots, n,\ l = 1, 2, 3 \\
\sum_{h=s_2+1}^{m} y_{jh}^{(1)} + x_{j1}^{(1)} + x_{j2}^{(1)} &\le a_1,\ \forall j = 1, \ldots, n \\
\sum_{h=s_1+1}^{s_2} y_{jh}^{(2)} + x_{j1}^{(2)} + x_{j3}^{(2)} &\le b_1,\ \forall j = 1, \ldots, n \\
\sum_{h=1}^{s_1} y_{jh}^{(3)} + x_{j2}^{(3)} + x_{j3}^{(3)} &\le c_1,\ \forall j = 1, \ldots, n \\
\sum_{j=1}^{n} y_{jh}^{(1)} &\le a_1,\ \forall h = s_2 + 1, \ldots, m \\
\sum_{j=1}^{n} y_{jh}^{(2)} &\le b_1,\ \forall h = s_1 + 1, \ldots, s_2 \\
\sum_{j=1}^{n} y_{jh}^{(3)} &\le c_1,\ \forall h = 1, \ldots, s_1 \\
\sum_{j=1}^{n} x_{jl}^{(1)} &= a_1,\ l = 1, 2 \\
\sum_{j=1}^{n} x_{jl}^{(2)} &= b_1,\ l = 1, 3 \\
\sum_{j=1}^{n} x_{jl}^{(3)} &= c_1,\ l = 2, 3 \\
\sum_{h=s_2+1}^{m} y_{jh}^{(1)} + x_{j1}^{(1)} + x_{j3}^{(1)} &\le \Delta(G_3) - b_1 - c_1 - r = a_2,\ \forall j = 1, \ldots, n \\
\sum_{h=1}^{s_1} y_{jh}^{(2)} + \sum_{h=s_2+1}^{m} y_{jh}^{(2)} + x_{j2}^{(2)} &\le \Delta(G_2) - a_1 - c_1 - r = b_2,\ \forall j = 1, \ldots, n \\
\sum_{h=s_1+1}^{s_2} y_{jh}^{(3)} + \sum_{h=s_2+1}^{m} y_{jh}^{(3)} + x_{j1}^{(3)} &\le \Delta(G_1) - a_1 - b_1 - r = c_2,\ \forall j = 1, \ldots, n
\end{aligned}$$

$$\begin{aligned}
\sum_{j=1}^{n} y_{jh}^{(1)} &\leq \Delta(G_3) - b_1 - c_1 - r = a_2,\ \forall h = 1, \ldots, s_2 \\
\sum_{j=1}^{n} y_{jh}^{(2)} &\leq \Delta(G_2) - a_1 - c_1 - r = b_2,\ \forall h = 1, \ldots, s_1, s_2 + 1, \ldots, m \\
\sum_{j=1}^{n} y_{jh}^{(3)} &\leq \Delta(G_1) - a_1 - b_1 - r = c_2,\ \forall h = s_1 + 1, \ldots, m \\
\sum_{j=1}^{n} x_{j3}^{(1)} &= \Delta(G_3) - b_1 - c_1 - r = a_2 \\
\sum_{j=1}^{n} x_{j2}^{(2)} &= \Delta(G_2) - a_1 - c_1 - r = b_2 \\
\sum_{j=1}^{n} x_{j1}^{(3)} &= \Delta(G_1) - a_1 - b_1 - r = c_2
\end{aligned}$$

$$\sum_{l=1}^{3} a_{jl} - \sum_{l=1}^{3} \sum_{i=1}^{3} x_{jl}^{(i)} \leq r,\ \forall j = 1, \ldots, n \tag{4.2}$$

$$\sum_{j=1}^{n} a_{jl} - \sum_{j=1}^{n} \sum_{i=1}^{3} x_{jl}^{(i)} = r,\ l = 1, 2, 3 \tag{4.3}$$

$$d(J_j) - \sum_{h=1}^{m} \sum_{i=1}^{3} y_{jh}^{(i)} \leq w,\ \forall j = 1, \ldots, n \tag{4.4}$$

$$d(P_h) - \sum_{j=1}^{n} \sum_{i=1}^{3} y_{jh}^{(i)} \leq w,\ \forall h = 1, \ldots, m \tag{4.5}$$

4.3 Preemptive open shop with four or more multiprocessors

The LP formulation for four multiprocessors becomes really large. First we give a possible form of the schedule. Therefore, we have a consistent form for the variables and constraints.

4.3.1 The normal form of a feasible schedule

We partition the processors into four sets of multiprocessors, denoted $G_1 = \{P_1, \ldots, s_1\}$, $G_2 = \{P_{s_1+1}, \ldots, P_{s_2}\}$, $G_3 = \{P_{s_2+1}, \ldots, P_{s_3}\}$ and $G_4 = \{P_{s_3+1}, \ldots, P_m\}$. Also, here we have $1 < s_1 < s_2 < s_3 < m$ and every number is an integer. If we

type of operations	interval
$(g,1),(i,2),(i,3),(i,4)$	(c_{11})
$(i,1),(g,2),(g,3),(g,4)$	(c_{12})
$(g,1),(g,2),(i,3),(i,4)$	(c_{21})
$(i,1),(i,2),(g,3),(g,4)$	(c_{22})
$(g,1),(i,2),(g,3),(i,4)$	(c_{31})
$(i,1),(g,2),(i,3),(g,4)$	(c_{32})
$(g,1),(i,2),(i,3),(g,4)$	(c_{41})
$(i,1),(g,2),(g,3),(i,4)$	(c_{42})
$(i,1),(g,2),(i,3),(i,4)$	(c_{51})
$(g,1),(i,2),(g,3),(g,4)$	(c_{52})
$(i,1),(i,2),(g,3),(i,4)$	(c_{61})
$(g,1),(g,2),(i,3),(g,4)$	(c_{62})
$(i,1),(i,2),(i,3),(g,4)$	(c_{71})
$(g,1),(g,2),(g,3),(i,4)$	(c_{72})
$(g,1),(g,2),(g,3),(g,4)$	(r)
$(i,1),(i,2),(i,3),(i,4)$	(w)

Table 4.3: The normal form with four multiprocessors.

have more then four, $p \geq 4$, multiprocessors then we continue in the same way, where $s_{p-1}+1$ will be the first processor in G_p.

In Figure 4.6 you see the template of the normal form. It is separated into 7 parts of type (C_i) and also (r) and (w). The ordering of the (C_i) is given by a complementary couple, (c_{i1}) and (c_{i2}). As in Figure 4.6, $(c_{11}) = (g,i,i,i)$ only has group operations on G_1 and $(c_{12}) = (i,g,g,g)$ contains group operations on G_2, G_3 and G_4. The symbol g describes the multiprocessors and its complementary is i, the individual processor. Thus, the order of the numeration of (C_i) is dependent only on (c_{i1}), because (c_{i2}) is then given, it is the complementary.

In Table 4.3, there we have all possible types of operations for each interval showing the normal form of the schedule.

We generalize this normal form of a schedule for more then four, say for p multiprocessors. There we have $(C_1),(C_2),\ldots,(C_{k_p})$. Still, (r) and (w) are always the same as in the case of four multiprocessors and the formula for k_p is given by equation (4.6). The total number of configurations, including (r) and (w), is given by 2^p. Because for every multiprocessor we have two possibilities: the multiprocessor is performing a group operation or it is not. It is not dependent on the number of processors because we execute only by groups of processors. That means if G_l is not performing group operations, then all P_h in G_l are performing

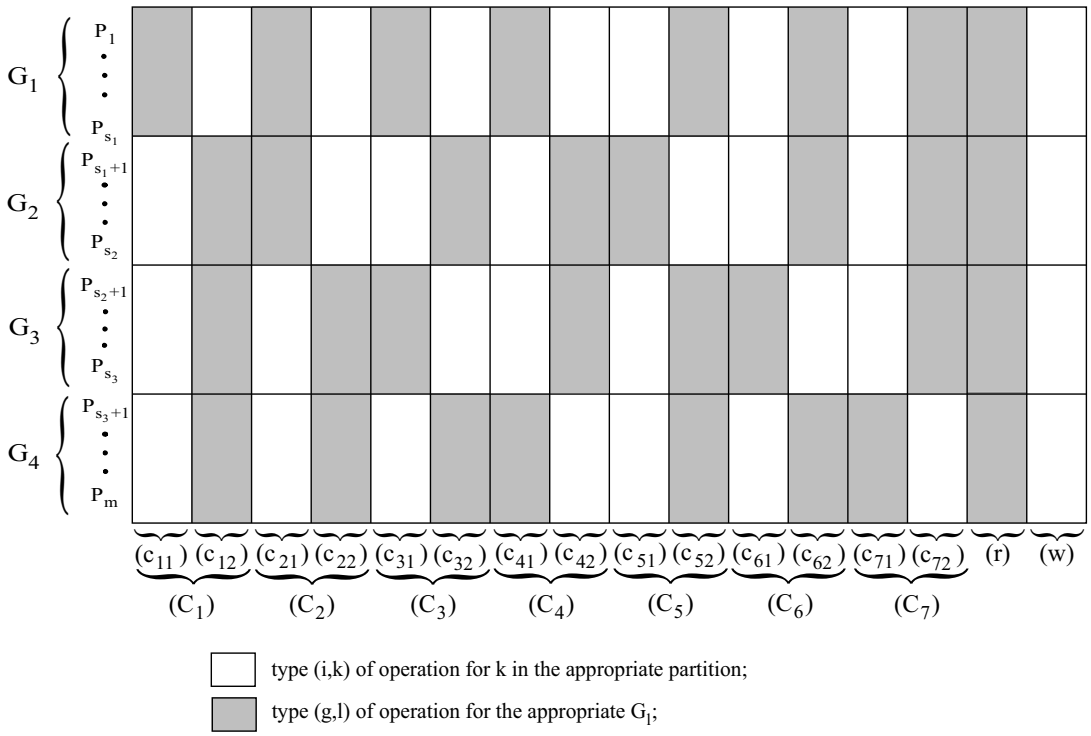

Figure 4.6: The normal form of a schedule with 4 multiprocessors.

individual operation or are paused. So the number of configurations is only dependent on p. If the number of the different configurations is 2^p then we have $2^p - 2$ intervals without (r) and (w). Moreover,

$$k_p = \frac{2^p - 2}{2} = 2^{p-1} - 1. \tag{4.6}$$

The order can be interpreted as a matrix $V_{k_p \times p}$ where v_{ik} is equal 1, if G_k is performing during (c_{i1}) and 0 if not. The case where all v_i are zero is represented as (w) and the complementary, where all v_i are 1, is (r).

Example 4.1. *We present the matrix V for the case of the normal form with four multiprocessors:*

$$V(p=4) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

and for five multiprocessors:

$$V(p=5) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}.$$

It is important to see which configurations are already given through the opposite interval (c_{i2}). And also those that are given through the symmetric configuration (only in the case where p is a even).

Furthermore, we present an idea to determine the constraints and give a formula to determine the number of constraints dependent on m, n and p. But first we have to introduce some variables necessary to construct the LP.

4.3.2 Variables

$x_{jl}^{(i)}$ **:** the amount of the group operation $\hat{O}_{jl}$ during the interval (C_i), where $i = 1, \ldots, k_p$;

$y_{jh}^{(i)}$ **:** the amount of the individual operation O_{jh} during the interval (C_i), where $i = 1, \ldots, k_p$;

c_{i1} **:** the length of the interval (c_{i1}), where $i = 1, \ldots, k_p$;

c_{i2} **:** the length of the interval (c_{i2}), where $i = 1, \ldots, k_p$;

r **:** the length of the overlap;

w : the length of the interval where only individual operations are processing;

4.3.3 Objective function

The objective function is to minimize the makespan, so we obtain the following objective

$$\min \big(\sum_{i=1}^{k_p} (c_{i1} + c_{i2}) + r + w \big).$$

In addition to this function we could lower the number of variables by one if we substituted

$$c_{k_p1} = \Delta(G_1) - r - c_{k_p2} - \sum_{i=2}^{k_p}(c_{i1} + c_{i2}).$$

For simplification of the notation this reductions is not used in the formulation of the constraints in the following descriptions.

4.3.4 Constraints

We give first the bound constraints and then the constraints of the different intervals C_i, where $i = 1, \ldots, k_p$.

Bound constraints

The first constraints are the bounds of time for all tasks of the individual and group operations. Logically they are not negative and no longer then a_{jl} or respectively b_{jh}. The inequations are the following:

$$0 \leq y_{jh}^{(i)} \leq b_{jh}, \ \forall j, h, i;$$

$$0 \leq x_{jh}^{(i)} \leq a_{jl}, \ \forall j, h, i;$$

Constraints for the interval (C_1)

We take (C_1) as the interval where (c_{11}) has only the first multiprocessor active and (c_{12}) has all other multiprocessor active because of the complementary of (c_{11}). For both, (c_{11}) and (c_{12}), we have to determine the total amount of jobs for each interval and we have to limit the processing time on each processor and multiprocessor by the interval lengths c_{11} or c_{12}.

There we have regarding the jobs for (c_{11})

$$\sum_{h=s_1+1}^{m} y_{jh}^{(1)} + x_{j1}^{(1)} \leq c_{11}, \ \forall j = 1, \ldots, n;$$

and for (c_{12})

$$\sum_{h=1}^{s_1} y_{jh}^{(1)} + \sum_{l=2}^{p} x_{jl}^{(1)} \leq c_{12}, \ \forall j = 1, \ldots, n;$$

That the processors do not exceed the interval length c_{11}, we limit them and get

$$\sum_{j=1}^{n} y_{jh}^{(1)} \leq c_{11}, \ \forall h = s_1 + 1, \ldots, m;$$

analog for the opposite configuration we have

$$\sum_{j=1}^{n} y_{jh}^{(1)} \leq c_{11}, \ \forall h = 1, \ldots, s_1;$$

for the multiprocessor G_1, there are exactly c_{11} time units of progressing. Therefore we receive in (c_{11})

$$\sum_{j=1}^{n} x_{j1}^{(1)} = c_{11};$$

and in (c_{12})

$$\sum_{j=1}^{n} x_{jl}^{(1)} = c_{12}, \ \forall l = 2, \ldots, p;$$

Constraints for the interval (C_i)

If we generalize the interval on its index, we no longer have a given order for which individual processors and multiprocessors are performing in each interval. So we call H_i the set of all processors, P_h, that are performing individual operations during (c_{i1}) and group operations during (c_{i2}), $\forall h \in H_i$. So we obtain the following constraints. Regarding the jobs for the interval (C_{i1}), we have

$$\sum_{h \in H_i} y_{jh}^{(i)} + \sum_{l | G_l \cap H_i = \emptyset} x_{jl}^{(i)} \leq c_{i1}, \ \forall j = 1, \ldots, n;$$

and for (c_{i2}) we have

$$\sum_{h \notin H_i} y_{jh}^{(i)} + \sum_{l | G_l \cap H_i \neq \emptyset} x_{jl}^{(i)} \leq c_{i2}, \ \forall j = 1, \ldots, n;$$

with respect to the processors P_h performing individual operations in (c_{i1}) and (c_{i2}), we have

$$\sum_{j=1}^{n} y_{jh}^{(i)} \leq c_{i1}, \ \forall h \in H_i$$

$$\sum_{j=1}^{n} y_{jh}^{(i)} \leq c_{i2}, \ \forall h \notin H_i$$

and for the multiprocessors, we have

$$\sum_{j=1}^{n} x_{jl}^{(i)} = c_{i1}, \ \forall \{l | G_l \cap H_i = \emptyset\};$$

$$\sum_{j=1}^{n} x_{jl}^{(i)} = c_{i2}, \ \forall \{l | G_l \cap H_i \neq \emptyset\};$$

Constraints for the interval (r)

For the overlap we have two constraints because we need none for the individual processes. One bounds the time for the jobs and the other is for the multiprocessor. For the jobs, we must see that the total time, where only group operations take place does not exceed r, so we obtain

$$\sum_{l=1}^{p} a_{jl} - \sum_{l=1}^{p} \sum_{i=1}^{k_p} x_{jl}^{(i)} \leq r, \ \forall j = 1, \ldots, n;$$

and analog for the multiprocessors, we know

$$\sum_{j=1}^{n} a_{jl} - \sum_{j=1}^{n} \sum_{i=1}^{k_p} x_{jl}^{(i)} = r, \ \forall l = 1, \ldots, p;$$

Constraints for the interval (w)

Finally, we also give the constraints for the last interval in the normal form of the schedule. It is the interval, where we have only individual processes, so we have only two constraints.

The constraint for the jobs is that the total time for the job J_j without the previous performances during any (C_i) do not exceed w. Therefore we obtain

$$d(J_j) - \sum_{h=1}^{m} \sum_{i=1}^{k_p} y_{jh}^{(i)} \leq w, \ \forall j = 1, \ldots, n;$$

and similarly for the processors, we have

$$d(P_h) - \sum_{j=1}^{n} \sum_{i=1}^{k_p} y_{jh}^{(i)} \leq w, \ \forall h = 1, \ldots, m;$$

4.3.5 Number of constraints

Here we will give an idea how to calculate the number of the necessary constraints depending on m, n and p.

We have seen that the construction of the constraints is built on the different configurations. If we know the number of different configurations then we know the number of constraints. For each configuration we must bound the job load and the processor load for single processors and multiprocessors. So for almost all configurations we have two times three constraints (because each configuration is a couple of two complementary time intervals), except for the special configurations of w and r where we have two. Because they do not have both types of operations, they have either individual or group operations. Hence we can say

we have $2 \times 3 \times k_p + 4$ types of constraints on the intervals. And each type of constraint has a multiplicity dependent on the range of the jobs, the active individual processors or multiprocessors. The multiplicity can be obtained from the LP (4.1) by the quantity of the $\forall$'s for each constraint. If we take as an example the last type of constraint the constraint (4.5), there we have a multiplicity of m, because we have to write this constraint for each individual processor. Therefore, we obtain for for any interval (C_i) $2n+m+p$ constraints, for (r) $n+p$ constraints and for (w) $n+m$ constraints.

On the other side we have bound constraints for each $x_{jl}^{(i)}$ and $y_{jh}^{(i)}$. Here we have $n \times k_p \times p$ bound constraints for the $x_{jl}^{(i)}$ and $n \times k_p \times m$ for the $y_{jh}^{(i)}$.

Finally we have $k_p \times (2n+m+p) + (n+p) + (n+m) + n \times k_p \times p + n \times k_p \times m$ constraints. Hence, we have the number

$$k_p \cdot n \cdot (p+m) + (2n+m+p)(k+p+1).$$

Chapter 5

An approximation algorithm for (PI) with a constant distance from (PF)

5.1 Introduction to the problem

In this chapter we search for a (PI) solution within a constant distance from an optimal (PF) solution. We were motivated by Asratian and de Werra [1]. In their paper they presented an approximation algorithm that gives us a timetable that is within $\frac{7}{6}$ of the maximum workload of any processor or any job under natural assumptions (see [1]). We define the workload of job J_j as

$$w(J_j) = \sum_{l=1}^{p} a_{jl} + \sum_{h=1}^{m} b_{jh},$$

and the workload of processor P_h as

$$w(P_h) = \sum_{j=1}^{n} a_{jl} + \sum_{j=1}^{n} b_{jh},\ h \in G_l.$$

Now we also can give a definition of the workload W,

$$W = \max\big(\max_j w(J_j), \max_h w(P_h)\big).$$

The makespan of a timetable is always integral, because we assume one lecture always takes integer time units. If we define the makespan of an optimal timetable as t^*, then this result can be stated as follows

$$\frac{t^*}{W} \leq \frac{7}{6}$$

or

$$t^* - W \leq \frac{1}{6} \cdot W$$

In this chapter we prove that there is an approximation algorithm for (PI) with fixed p such that $t^* - t \leq cte$ for some constant cte where t is the optimal fractional makespan.

Our motivation for such an approximation algorithm is based on [13]. There Kubiak has shown a similar algorithm for $p = 2$. We intend to compute the (PF) solution, then with the rounding network we make a rounding for certain tasks in the intervals (C_i). Finally we use the algorithm of Gonzales and Sahni or a edge coloring algorithm to obtain the schedule. Now, we try to find such an algorithm for the case with three multiprocessors and finally for any $p \geq 3$. First, we will introduce the rounding network (RN) and then with respect to the rounding network, we will prove the existence of such a constant.

In the next section we give a description of (RN).

5.2 The rounding network

We introduce a rounding network (RN) for the intervals $(a_1), (a_2), (b_1), (b_2), (c_1)$ and (c_2), this part is denoted as $\mathcal{A}$, which will be used to round the LP (4.1) solution in such a way that we can find a constant deviation from the (PF) solution. That means, the LP gets all the values of all the variables $x_{jl}^{(i)}, y_{jh}^{(i)}$ and of course the lengths of $a_1, a_2, b_1, b_2, c_1, c_2, r$ and w. But some of these variables do not have to be integers.

Now the (RN) shows us, which task of operation in $\mathcal{A}$ is rounded up or down. The rounding changes the makespan for $\mathcal{A}$, and it produces a leftover. An leftover represent the amount of operations that are not performed during $\mathcal{A}$ and have to be done in (r) or (w).

The aim is to show that the change of the optimal makespan for $\mathcal{A}$ and the leftover is not more then a constant number of time units for a fixed p. We will later see it is 21 for $p = 3$. But first, let's take a look on the body structure of (RN) and its upper and lower bounds.

5.2.1 The upper an lower bounds of (RN)

At the beginning we give a map, Figure 5.1, of the body structure parts in Figures 5.2-5.6. Because of its size the network (RN) is given in five figures. The Figures 5.4,5.5 and 5.6 describe the body and the sink t. The Figures 5.2 and 5.3 form the head with the source s. The bounds are written as paires ("lower bound", "upper bound"). A precise list containing the bounds is included in the appendix A.

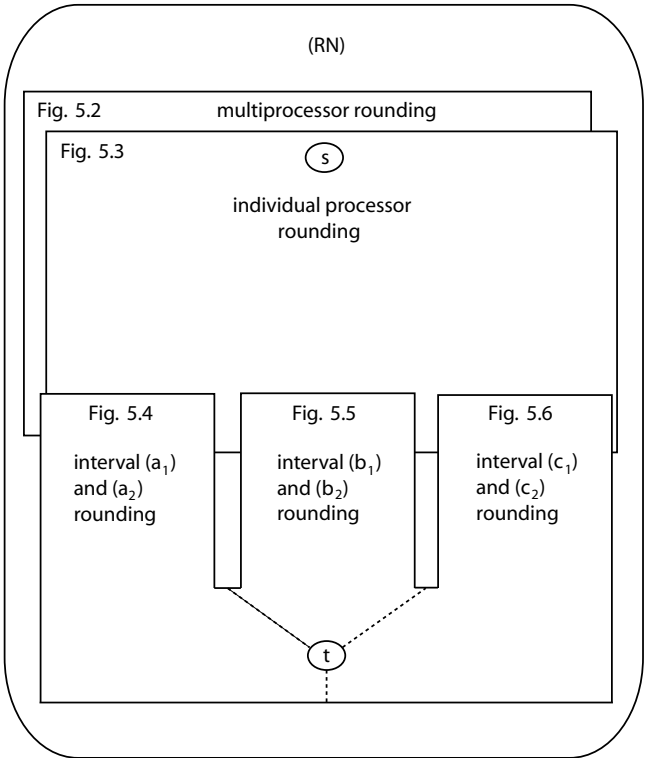

Figure 5.1: The body structure of (RN).

The network is split into different Figures but the same vertices have the same notations.

Furthermore, we introduce three new indices h_1, h_2 and h_3, where $h_1 \in G_1, h_2 \in G_2$ and $h_3 \in G_3$. If we have $y_{jh_2}^{(i)}$ then we know the operation is performed on one of the individual processors in the set of G_2 is $P_{s_1+1}, \ldots, P_{s_2-1}$ or P_{s_2}.

The bounds for μ and λ are necessary to bound the total processing time for each individual processor P_h (given by λ_h) and for each multiprocessor G_l (given by μ_l) in $\mathcal{A}$, is maximally one time unit longer.

$$\mu_l = (\lfloor \sum_{i=1}^{3} \sum_{j=1}^{n} x_{jl}^{(i)} \rfloor, \lceil \sum_{i=1}^{3} \sum_{j=1}^{n} x_{jl}^{(i)} \rceil), \ \forall l = 1, 2, 3; \tag{5.1}$$

$$\lambda_h = (\lfloor \sum_{i=1}^{3} \sum_{j=1}^{n} y_{jh}^{(i)} \rfloor, \lceil \sum_{i=1}^{3} \sum_{j=1}^{n} y_{jh}^{(i)} \rceil), \ \forall h = 1, \ldots, m; \tag{5.2}$$

The bounds in (RN) represented by α assure us that each multiprocessor in every interval of $\mathcal{A}$ does not exceed more then one unit and β is the analog for

each individual processor.

$$\alpha_l^i = (\lfloor \sum_{j=1}^{n} x_{jl}^{(i)} \rfloor, \lceil \sum_{j=1}^{n} x_{jl}^{(i)} \rceil),\ \forall l = 1,2,3,\ i = 1,2,3; \tag{5.3}$$

$$\beta_h^i = (\lfloor \sum_{j=1}^{n} y_{jh}^{(i)} \rfloor, \lceil \sum_{j=1}^{n} y_{jh}^{(i)} \rceil),\ \forall h = 1,\ldots,m,\ i = 1,2,3; \tag{5.4}$$

The bounds in (RN) represented by θ and η concern the rounding for each processing time in every interval of $\mathcal{A}$. Here, the network satisfies that every processing time is an integer. And it tells us if we have to round up or down each task included in $x_{jl}^{(i)}$ by θ. Similarly for $y_{jh}^{(i)}$ we have the bounds given by η.

$$\begin{aligned} \theta_{jl}^i &= (\lfloor x_{jl}^{(i)} \rfloor, \lceil x_{jl}^{(i)} \rceil),\ \forall j = 1,\ldots,n,\ l = 1,2,3,\ i = 1,2,3; \\ \eta_{jh}^i &= (\lfloor y_{jh}^{(i)} \rfloor, \lceil y_{jh}^{(i)} \rceil),\ \forall j = 1,\ldots,n,\ h = 1,\ldots,m,\ i = 1,2,3; \end{aligned} \tag{5.5}$$

Now we present the bounds for σ, which assures that we round balanced all processing times concerning job J_j in the appropriate interval of $\mathcal{A}$.

$$\begin{aligned}
\sigma_1^1 &= (\lfloor x_{j1}^{(1)} + x_{12}^{(1)} + \sum_{h_3 \in G_3} y_{1h_3}^{(1)} \rfloor, \lceil x_{11}^{(1)} + x_{12}^{(1)} + \sum_{h_3 \in G_3} y_{1h_3}^{(1)} \rceil) \\
\sigma_1^2 &= (\lfloor x_{13}^{(1)} + \sum_{h_1 \in G_1} y_{1h_1}^{(1)} + \sum_{h_2 \in G_2} y_{1h_2}^{(1)} \rfloor, \lceil x_{13}^{(1)} + \sum_{h_1 \in G_1} y_{1h_1}^{(1)} + \sum_{h_2 \in G_2} y_{1h_2}^{(1)} \rceil) \\
\sigma_1^3 &= (\lfloor x_{11}^{(2)} + x_{13}^{(2)} + \sum_{h_2 \in G_2} y_{1h_2}^{(2)} \rfloor, \lceil x_{11}^{(2)} + x_{13}^{(2)} + \sum_{h_2 \in G_2} y_{1h_2}^{(2)} \rceil) \\
\sigma_1^4 &= (\lfloor x_{12}^{(2)} + \sum_{h_1 \in G_1} y_{1h_1}^{(2)} + \sum_{h_3 \in G_3} y_{1h_3}^{(2)} \rfloor, \lceil x_{12}^{(2)} + \sum_{h_1 \in G_1} y_{1h_1}^{(2)} + \sum_{h_3 \in G_3} y_{1h_3}^{(2)} \rceil) \\
\sigma_j^1 &= (\lfloor x_{j1}^{(1)} + x_{j2}^{(1)} + \sum_{h_3 \in G_3} y_{jh_3}^{(1)} \rfloor, \lceil x_{j1}^{(1)} + x_{j2}^{(1)} + \sum_{h_3 \in G_3} y_{jh_3}^{(1)} \rceil) \\
\sigma_j^2 &= (\lfloor x_{j3}^{(1)} + \sum_{h_1 \in G_1} y_{jh_1}^{(1)} + \sum_{h_3 \in G_3} y_{jh_3}^{(1)} \rfloor, \lceil x_{j3}^{(1)} + \sum_{h_1 \in G_1} y_{jh_1}^{(1)} + \sum_{h_3 \in G_3} y_{jh_3}^{(1)} \rceil) \\
\sigma_j^3 &= (\lfloor x_{j1}^{(2)} + x_{j3}^{(2)} + \sum_{h_2 \in G_2} y_{jh_2}^{(2)} \rfloor, \lceil x_{j1}^{(2)} + x_{j3}^{(2)} + \sum_{h_2 \in G_2} y_{jh_2}^{(2)} \rceil) \\
\sigma_j^4 &= (\lfloor x_{j2}^{(2)} + \sum_{h_1 \in G_1} y_{jh_1}^{(2)} + \sum_{h_3 \in G_3} y_{jh_3}^{(2)} \rfloor, \lceil x_{j2}^{(2)} + \sum_{h_1 \in G_1} y_{jh_1}^{(2)} + \sum_{h_3 \in G_3} y_{jh_3}^{(2)} \rceil)
\end{aligned}$$

$$\begin{aligned}
\sigma_j^5 &= (\lfloor x_{j2}^{(3)} + x_{j3}^{(3)} + \sum_{h_1 \in G_1} y_{jh_1}^{(3)} \rfloor, \lceil x_{j2}^{(3)} + x_{j3}^{(3)} + \sum_{h_1 \in G_1} y_{jh_1}^{(3)} \rceil) \\
\sigma_j^6 &= (\lfloor x_{j1}^{(3)} + \sum_{h_2 \in G_2} y_{jh_2}^{(3)} + \sum_{h_3 \in G_3} y_{jh_3}^{(3)} \rfloor, \lceil x_{j1}^{(3)} + \sum_{h_2 \in G_2} y_{jh_2}^{(3)} + \sum_{h_3 \in G_3} y_{jh_3}^{(3)} \rceil) \\
\sigma_n^3 &= (\lfloor x_{n1}^{(2)} + x_{n3}^{(2)} + \sum_{h_2 \in G_2} y_{nh_2}^{(2)} \rfloor, \lceil x_{n1}^{(2)} + x_{n3}^{(2)} + \sum_{h_2 \in G_2} y_{nh_2}^{(2)} \rceil) \\
\sigma_n^4 &= (\lfloor x_{n2}^{(2)} + \sum_{h_1 \in G_1} y_{nh_1}^{(2)} + \sum_{h_3 \in G_3} y_{nh_3}^{(2)} \rfloor, \lceil x_{n2}^{(2)} + \sum_{h_1 \in G_1} y_{nh_1}^{(2)} + \sum_{h_3 \in G_3} y_{nh_3}^{(2)} \rceil) \\
\sigma_n^5 &= (\lfloor x_{n2}^{(3)} + x_{n3}^{(3)} + \sum_{h_1 \in G_1} y_{nh_1}^{(3)} \rfloor, \lceil x_{n2}^{(3)} + x_{n3}^{(3)} + \sum_{h_1 \in G_1} y_{nh_1}^{(3)} \rceil) \\
\sigma_n^6 &= (\lfloor x_{n1}^{(3)} + \sum_{h_2 \in G_2} y_{nh_2}^{(3)} + \sum_{h_3 \in G_3} y_{nh_3}^{(3)} \rfloor, \lceil x_{n1}^{(3)} + \sum_{h_2 \in G_2} y_{nh_2}^{(3)} + \sum_{h_3 \in G_3} y_{nh_3}^{(3)} \rceil)
\end{aligned}$$

And finally we introduce the bounds for γ. γ is the bound that gives us the bound for the total processing time in $\mathcal{A}$ with respect to each job.

$$\gamma_j = (\lfloor \sum_{i=1}^{3} \sum_{l=1}^{3} x_{jl}^{(i)} + \sum_{i=1}^{3} \sum_{h=1}^{m} y_{jh}^{(i)} \rfloor, \lceil \sum_{i=1}^{3} \sum_{l=1}^{3} x_{jl}^{(i)} + \sum_{i=1}^{3} \sum_{h=1}^{m} y_{jh}^{(i)} \rceil), \ \forall j = 1, \ldots, n; \tag{5.6}$$

5.2.2 Existence of a feasible flow in (RN)

That the (RN) contains a feasible flow, is given by the following theorem.

Theorem 5.1. *The rounding network (RN) contains a feasible flow.*

Proof. Consider the values for the optimal fractional makespan. They present in (RN) a feasible flow because

$$\begin{aligned}
\lfloor x_{jl}^{(i)} \rfloor \leq x_{jl}^{(i)} \leq \lceil x_{jl}^{(i)} \rceil, \ \forall j, l, i, \\
\lfloor y_{jh}^{(i)} \rfloor \leq y_{jh}^{(i)} \leq \lceil y_{jh}^{(i)} \rceil, \ \forall j, h, i.
\end{aligned}$$

Every upper bound is the rounding up of the optimal makespan for the fractional model; the lower bound is the rounding down. Hence, we have a feasible flow on every arc, so we have a feasible flow in (RN). □

5.3 A constant error

Here we try to verify the assumption that for the open shop with integer preemptions and three multiprocessors, we can find a constant in such a way, that the optimal integer makespan has a constant deviation from the optimal fractional makespan. We formulate this fact in the following theorem.

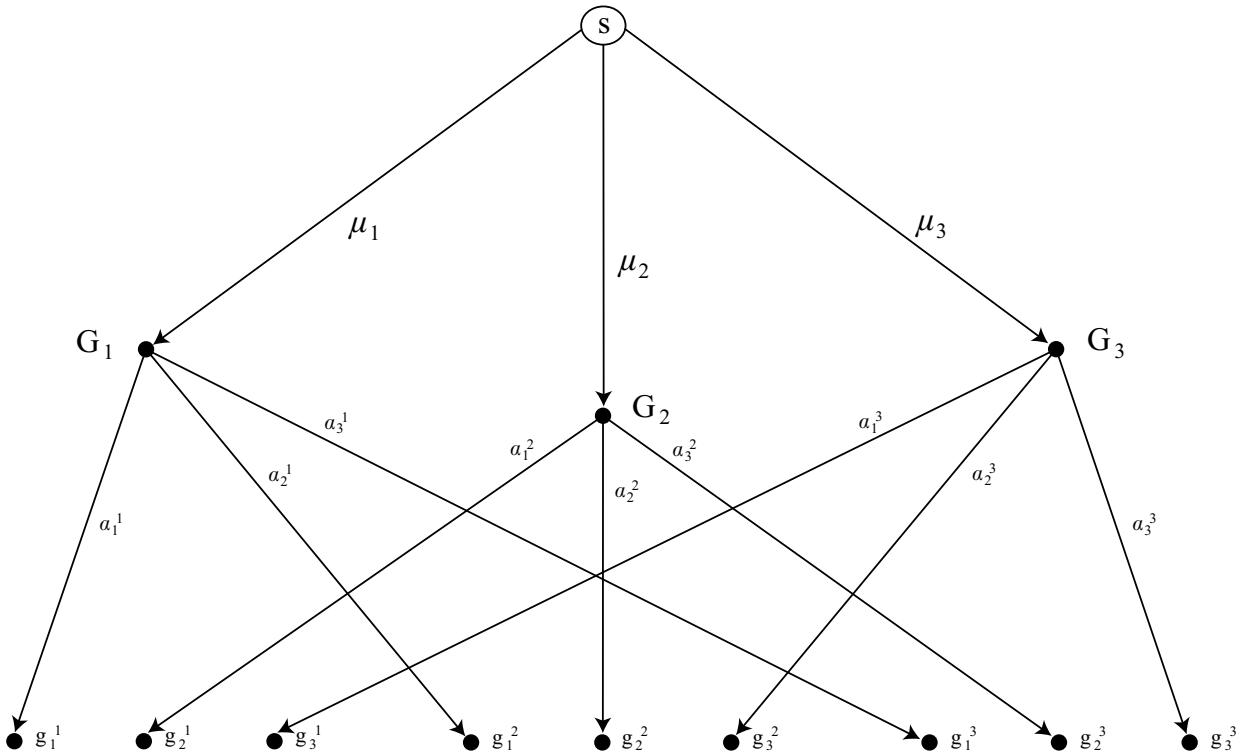

Figure 5.2: The head of the rounding network with bounds on each multiprocessor workload.

Theorem 5.2. *The solution obtained by a rounding network, denote it by t^{RN}, for an open shop with integer preemptions and three multiprocessors is not longer then 21 time units then the optimal makespan t for the fractional model,*

$$t^{RN} - t \leq 21.$$

Proof. We know there is a feasible flow in (RN) due to the Theorem 5.1. Since all upper and lower bounds are integral, then there exists a maximal flow in (RN), that is integral by the integral flow Theorem 2.5. To prove the theorem we consider the rounding network (RN). This network gives us a rounding, which rounds each interval of $\mathcal{A}$ up or down by at most one time unit. The individual flow in the arcs (P_h^i, J_j^*) is denoted by $e_{jh}^{(i)}$ and similarly for the group flow in (g_l^i, J_j^*) is denoted by $d_{jl}^{(i)}$. Here J_j^* is the adequate vertex for the convenient interval in $\mathcal{A}$. We define $t_{\mathcal{A}}^*$ as the makespan during $\mathcal{A}$ in the integer and $t_{\mathcal{A},J}$ the makespan in the fractional case, with respect to each job J_j and $t_{\mathcal{A},P}$ with respect to the processors.

The proof holds due to the König's edge coloring Theorem and the following Lemmas 5.1-5.6. The Lemma 5.1 assures us that the rounding with respect to each job J_j, given through the network, will not exceed more then one time unit. By Lemma 5.2 we show that the leftover for (r), caused by the rounding,

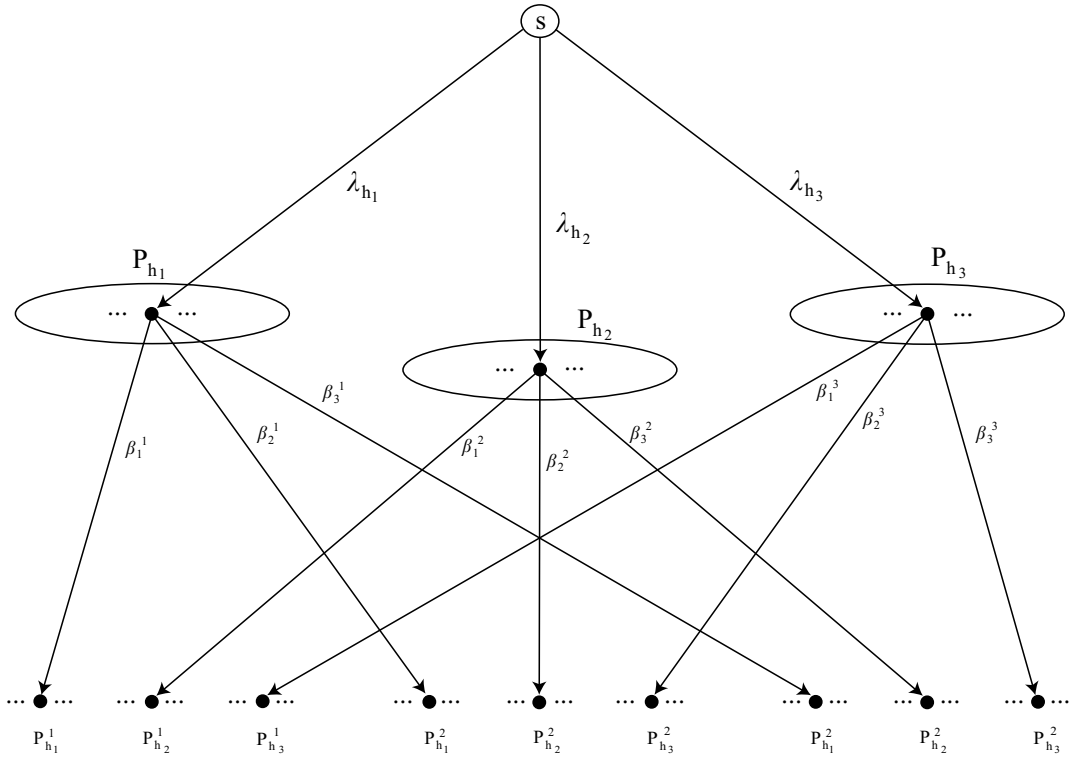

Figure 5.3: The head of the rounding network with bounds on each individual processor workload.

is also bounded. Similarly we have the bound for the leftover in (w), given by Lemma 5.3. The Lemmas 5.4-5.6 give us the maximal deviation for each individual processor and each multiprocessor. Lemma 5.4 says the flow in the network increases the makespan in $\mathcal{A}$ not more then two time units. The leftover for (r), for each multiprocessor, is given by Lemma 5.5. Similarly, the leftover for each individual processor is given by Lemma 5.6.

Lemma 5.1. *$\forall j$, we have $t^*_{\mathcal{A}} \leq t_{\mathcal{A},J} + 1$.*

Proof. Due to the network's bound γ (see equation (5.6)), we have

$$\begin{aligned}
\sum_{i=1}^{3}\sum_{h=1}^{m} e_{jh}^{(i)} + \sum_{i=1}^{3}\sum_{l=1}^{3} d_{jl}^{(i)} &\leq \lceil \gamma_j \rceil = \lceil \sum_{i=1}^{3}\sum_{h=1}^{m} y_{jh}^{(i)} + \sum_{i=1}^{3}\sum_{l=1}^{3} x_{jl}^{(i)} \rceil \\
&\leq \sum_{i=1}^{3}\sum_{h=1}^{m} y_{jh}^{(i)} + \sum_{i=1}^{3}\sum_{l=1}^{3} x_{jl}^{(i)} + 1 \\
t^*_{\mathcal{A}} &\leq t_{\mathcal{A},J} + 1
\end{aligned}$$

□

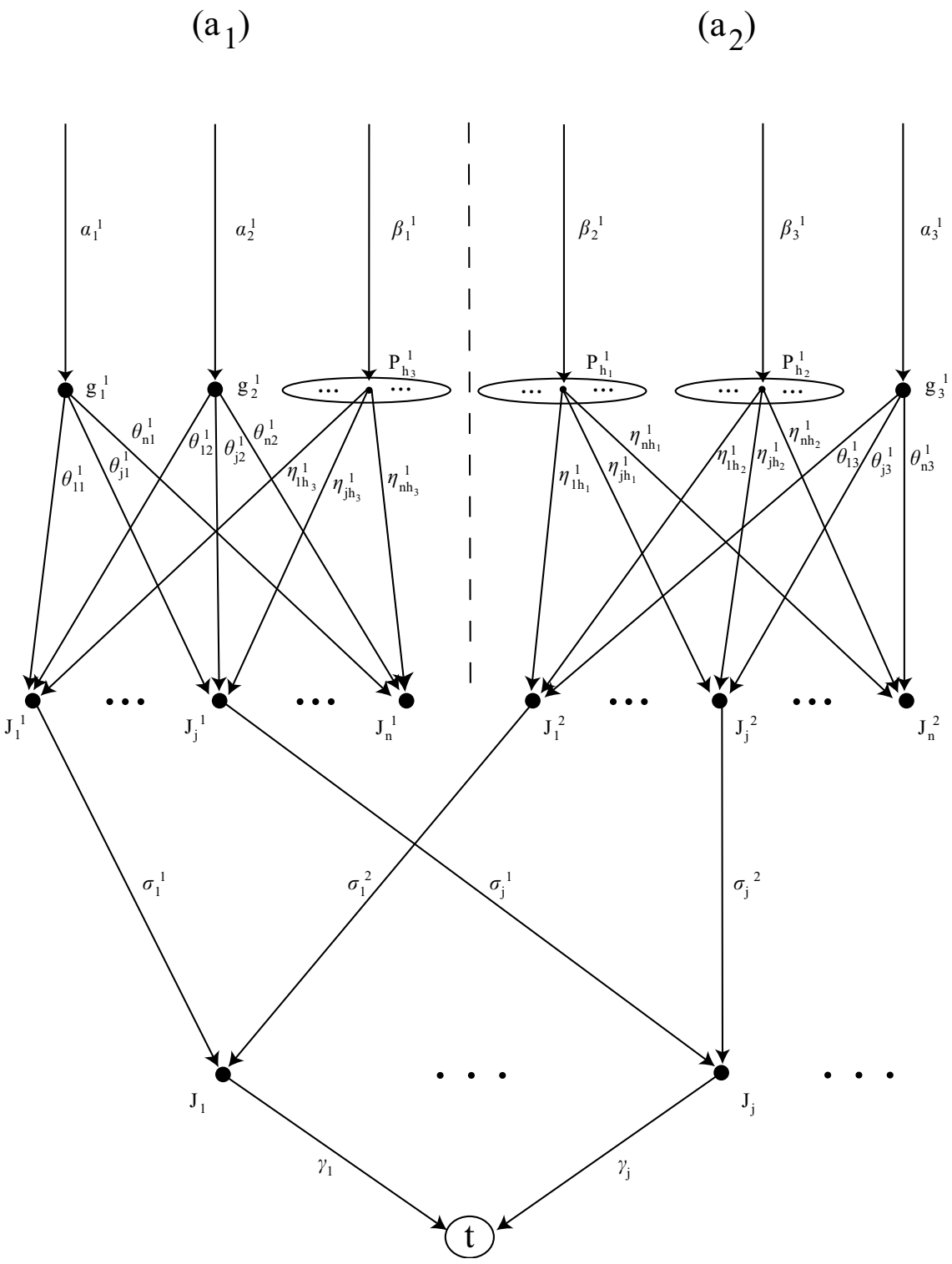

Figure 5.4: The rounding network in interval (a_1) and (a_2).

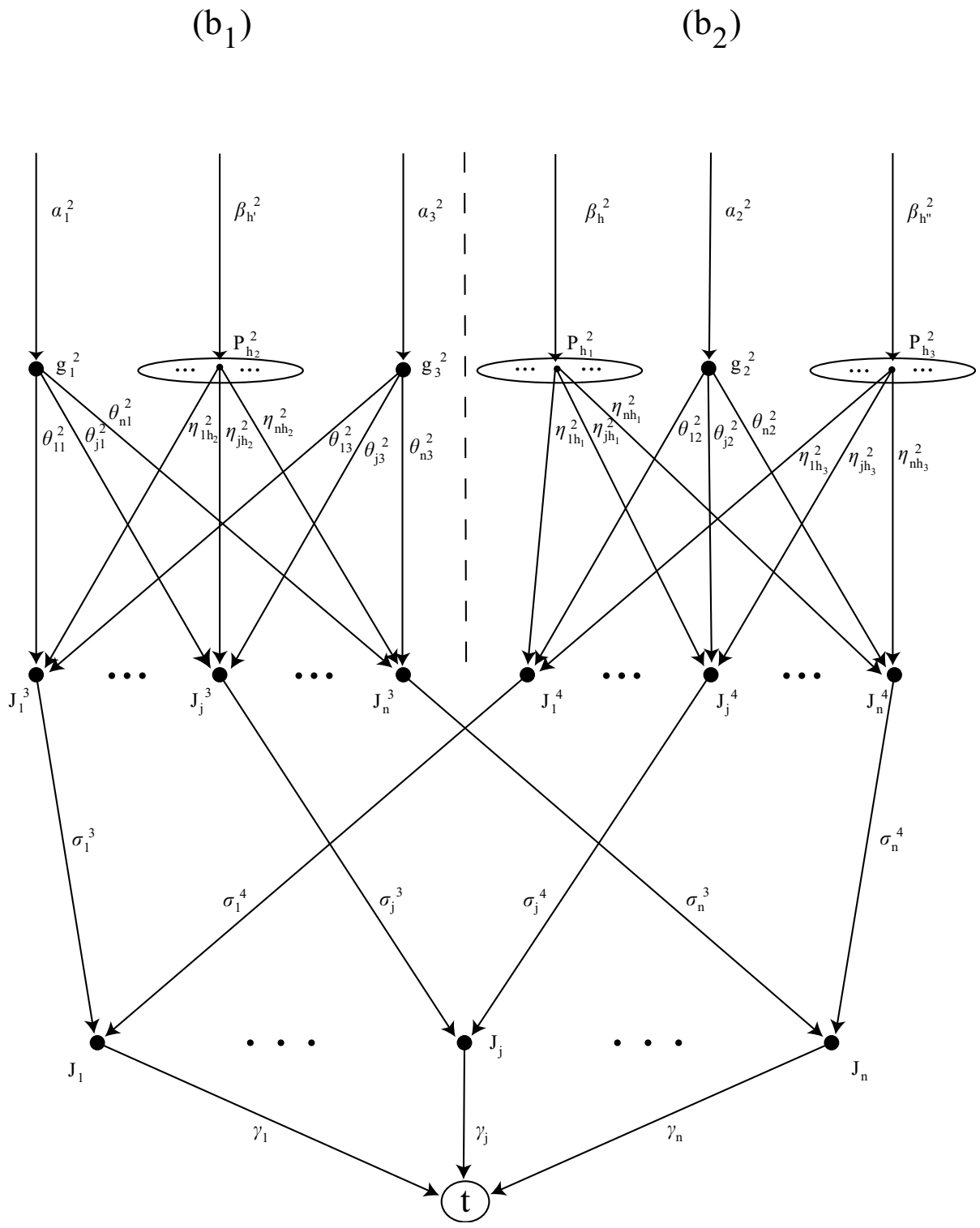

Figure 5.5: The rounding network in interval (b_1) and (b_2).

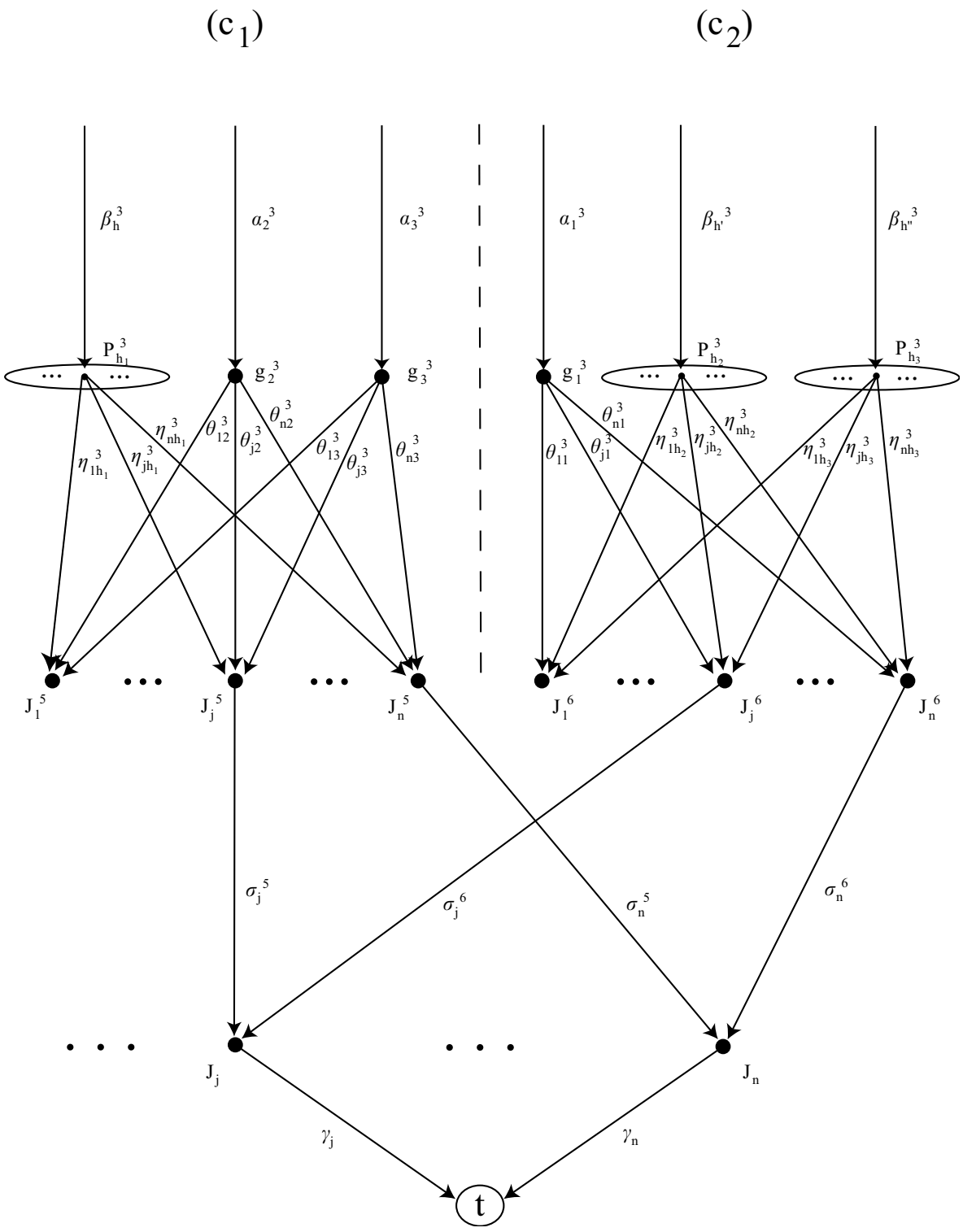

Figure 5.6: The rounding network in interval (c_1) and (c_2).

Lemma 5.2. *$\forall j$, we have*

$$\sum_{j=1}^{n} a_{jl} - \sum_{i=1}^{3}\sum_{l=1}^{3} d_{jl}^{(i)} \leq \lceil r \rceil + 8.$$

Proof. If we define $x_{jl}^{(i)} = \lfloor x_{jl}^{(i)} \rfloor + \epsilon_{jl}^{(i)},\ 0 \leq \epsilon_{jl}^{(i)} < 1,\ \forall j, l, i$ and with the LP equation (4.2), we know

$$\begin{aligned} r \geq \sum_{j=1}^{n} a_{jl} - \sum_{i=1}^{3}\sum_{l=1}^{3} x_{jl}^{(i)} &= \sum_{j=1}^{n} a_{jl} - \sum_{i=1}^{3}\sum_{l=1}^{3} \lfloor x_{jl}^{(i)} \rfloor - \sum_{i=1}^{3}\sum_{l=1}^{3} \epsilon_{jl}^{(i)} \\ &\geq \sum_{j=1}^{n} a_{jl} - \sum_{i=1}^{3}\sum_{l=1}^{3} d_{jl}^{(i)} - \sum_{i=1}^{3}\sum_{l=1}^{3} \epsilon_{jl}^{(i)} \end{aligned}$$

The leftover of group operations to do in (r) is then

$$\begin{aligned} \sum_{j=1}^{n} a_{jl} - \sum_{i=1}^{3}\sum_{l=1}^{3} d_{jl}^{(i)} &\leq \lfloor r + \sum_{i=1}^{3}\sum_{l=1}^{3} \epsilon_{jl}^{(i)} \rfloor \\ &\leq \lceil r \rceil + \lfloor \sum_{i=1}^{3}\sum_{l=1}^{3} \epsilon_{jl}^{(i)} \rfloor \\ &\leq \lceil r \rceil + 8. \end{aligned}$$

□

Lemma 5.3. *$\forall j$, we have*

$$d(J_j) - \sum_{i=1}^{3}\sum_{h=1}^{m} e_{jh}^{(i)} \leq \lceil w \rceil + 9.$$

Proof. If we take the entire flow in $\mathcal{A}$, then we know $\forall j$ due to the γ bound (5.6) from the network is

$$\begin{aligned} \sum_{i=1}^{3}\sum_{h=1}^{m} e_{jh}^{(i)} + \sum_{i=1}^{3}\sum_{l=1}^{3} d_{jl}^{(i)} &\geq \lfloor \gamma_j \rfloor = \lfloor \sum_{i=1}^{3}\sum_{h=1}^{m} y_{jh}^{(i)} + \sum_{i=1}^{3}\sum_{l=1}^{3} x_{jl}^{(i)} \rfloor \\ &\geq \lfloor \sum_{i=1}^{3}\sum_{h=1}^{m} y_{jh}^{(i)} \rfloor + \lfloor \sum_{i=1}^{3}\sum_{l=1}^{3} x_{jl}^{(i)} \rfloor \end{aligned} \tag{5.7}$$

From the LP equation (4.4), we have

$$\sum_{i=1}^{3}\sum_{h=1}^{m} y_{jh}^{(i)} \geq d(J_j) - w,\ \forall j$$

and if we round down, we obtain

$$\lfloor \sum_{i=1}^{3} \sum_{h=1}^{m} y_{jh}^{(i)} \rfloor \geq d(J_j) - \lceil w \rceil \tag{5.8}$$

The bound θ (5.5) of the network gives us

$$\lceil x_{jl}^{(i)} \rceil \geq d_{jl}^{(i)} \geq \lfloor x_{jl}^{(i)} \rfloor, \ \forall j, l, i.$$

Furthermore, we can show

$$\begin{aligned} x_{jl}^{(i)} &= \lceil x_{jl}^{(i)} \rceil - \overline{\epsilon}_{jl}^{(i)}, \ 0 \leq \overline{\epsilon}_{jl}^{(i)} < 1, \ \forall j, l, i \\ \sum_{i=1}^{3} \sum_{l=1}^{3} x_{jl}^{(i)} &= \sum_{i=1}^{3} \sum_{l=1}^{3} \lceil x_{jl}^{(i)} \rceil - \sum_{i=1}^{3} \sum_{l=1}^{3} \overline{\epsilon}_{jl}^{(i)} \\ \lfloor \sum_{i=1}^{3} \sum_{l=1}^{3} x_{jl}^{(i)} \rfloor &= \sum_{i=1}^{3} \sum_{l=1}^{3} \lceil x_{jl}^{(i)} \rceil - \lceil \sum_{i=1}^{3} \sum_{l=1}^{3} \overline{\epsilon}_{jl}^{(i)} \rceil \end{aligned} \tag{5.9}$$

Finally, we take the entire flow in $\mathcal{A}$ and due to the inequalities (5.7)-(5.9), we have

$$\begin{aligned} \sum_{i=1}^{3} \sum_{h=1}^{m} e_{jh}^{(i)} + \sum_{i=1}^{3} \sum_{l=1}^{3} d_{jl}^{(i)} &\geq \lfloor \sum_{i=1}^{3} \sum_{h=1}^{m} y_{jh}^{(i)} \rfloor + \lfloor \sum_{i=1}^{3} \sum_{l=1}^{3} x_{jl}^{(i)} \rfloor \\ &\geq d(J_j) - \lceil w \rceil + \lfloor \sum_{i=1}^{3} \sum_{l=1}^{3} x_{jl}^{(i)} \rfloor \\ &\geq d(J_j) - \lceil w \rceil + \sum_{i=1}^{3} \sum_{l=1}^{3} \lceil x_{jl}^{(i)} \rceil - \lceil \sum_{i=1}^{3} \sum_{l=1}^{3} \overline{\epsilon}_{jl}^{(i)} \rceil \\ &\geq d(J_j) - \lceil w \rceil + \sum_{i=1}^{3} \sum_{l=1}^{3} d_{jl}^{(i)} - \lceil \sum_{i=1}^{3} \sum_{l=1}^{3} \overline{\epsilon}_{jl}^{(i)} \rceil \\ \lceil w \rceil + \lceil \sum_{i=1}^{3} \sum_{l=1}^{3} \overline{\epsilon}_{jl}^{(i)} \rceil &\geq d(J_j) - \sum_{i=1}^{3} \sum_{h=1}^{m} e_{jh}^{(i)} \\ \lceil w \rceil + 9 &\geq d(J_j) - \sum_{i=1}^{3} \sum_{h=1}^{m} e_{jh}^{(i)} \end{aligned}$$

□

Now, we must proof that we can bound the leftover also with respect to each multiprocessor and each individual processor.

Lemma 5.4. *For every processor we have*

$$t^*_{\mathcal{A}} \leq t_{\mathcal{A},P} + 2$$

.

Proof. Due to the bounds μ (5.1) and λ (5.2) in (RN), we have $\forall h, l$

$$\begin{aligned} t^*_{\mathcal{A}} = \sum_{i=1}^{3}\sum_{j=1}^{n} e^{(i)}_{jh} + \sum_{i=1}^{3}\sum_{j=1}^{n} d^{(i)}_{jl} &\leq \lceil \sum_{i=1}^{3}\sum_{j=1}^{n} y^{(i)}_{jh} \rceil + \lceil \sum_{i=1}^{3}\sum_{j=1}^{n} x^{(i)}_{jl} \rceil \\ &\leq \lfloor \sum_{i=1}^{3}\sum_{j=1}^{n} y^{(i)}_{jh} \rfloor + \lfloor \sum_{i=1}^{3}\sum_{j=1}^{n} x^{(i)}_{jl} \rfloor + 2 \\ &\leq \sum_{i=1}^{3}\sum_{j=1}^{n} y^{(i)}_{jh} + \sum_{i=1}^{3}\sum_{j=1}^{n} x^{(i)}_{jl} + 2 \\ &\leq t_{\mathcal{A},P} + 2 \end{aligned}$$

□

Lemma 5.5. *$\forall G_l$, we have*

$$\sum_{j=1}^{n} a_{jl} - \sum_{i=1}^{3}\sum_{j=1}^{n} d^{(i)}_{jl} \leq \lfloor r \rfloor + 3.$$

Proof. Due to the bounds α (5.3) we have

$$\lfloor \sum_{j=1}^{n} x^{(i)}_{jl} \rfloor \leq \sum_{j=1}^{n} d^{(i)}_{jl} \leq \lceil \sum_{j=1}^{n} x^{(i)}_{jl} \rceil \tag{5.10}$$

Furthermore we can show that

$$\sum_{i=1}^{3}\sum_{j=1}^{n} x^{(i)}_{jl} \leq \sum_{i=1}^{3} \lceil \sum_{j=1}^{n} x^{(i)}_{jl} \rceil \leq \sum_{i=1}^{3} \lfloor \sum_{j=1}^{n} x^{(i)}_{jl} \rfloor + 3 \tag{5.11}$$

Now with the equations (5.10), (5.11) and the LP equation (4.3), we show

$$\begin{aligned} r = \sum_{j=1}^{n} a_{jl} - \sum_{i=1}^{3}\sum_{j=1}^{n} x^{(i)}_{jl} &\geq \sum_{j=1}^{n} a_{jl} - \sum_{i=1}^{3} \lceil \sum_{j=1}^{n} x^{(i)}_{jl} \rceil \\ &\geq \sum_{j=1}^{n} a_{jl} - \sum_{i=1}^{3} \lfloor \sum_{j=1}^{n} x^{(i)}_{jl} \rfloor - 3 \\ &\geq \sum_{j=1}^{n} a_{jl} - \sum_{i=1}^{3}\sum_{j=1}^{n} d^{(i)}_{jl} - 3 \end{aligned}$$

Finally, we obtain

$$r + 3 \geq \lfloor r \rfloor + 3 \geq \sum_{j=1}^{n} a_{jl} - \sum_{i=1}^{3} \sum_{j=1}^{n} d_{jl}^{(i)}.$$

□

Lemma 5.6. *$\forall P_h$, we have*

$$d(P_h) - \sum_{i=1}^{3} \sum_{j=1}^{n} e_{jh}^{(i)} \leq \lfloor w \rfloor + 3.$$

Proof. Due to the bounds β (5.4), we have

$$\lfloor \sum_{j=1}^{n} y_{jh}^{(i)} \rfloor \leq \sum_{j=1}^{n} e_{jh}^{(i)} \leq \lceil \sum_{j=1}^{n} y_{jh}^{(i)} \rceil \tag{5.12}$$

Then we know that

$$\sum_{i=1}^{3} \sum_{j=1}^{n} y_{jh}^{(i)} \leq \sum_{i=1}^{3} \lceil \sum_{j=1}^{n} y_{jh}^{(i)} \rceil \leq \sum_{i=1}^{3} \lfloor \sum_{j=1}^{n} y_{jh}^{(i)} \rfloor + 3 \tag{5.13}$$

Now we show, including the equations (5.12), (5.13) and the LP equation (4.5),

$$\begin{aligned} w \geq d(P_h) - \sum_{i=1}^{3} \sum_{j=1}^{n} y_{jh}^{(i)} &\geq d(P_h) - \sum_{i=1}^{3} \lceil \sum_{j=1}^{n} y_{jh}^{(i)} \rceil \\ &\geq d(P_h) - \sum_{i=1}^{3} \lfloor \sum_{j=1}^{n} y_{jh}^{(i)} \rfloor - 3 \\ &\geq d(P_h) - \sum_{i=1}^{3} \sum_{j=1}^{n} e_{jh}^{(i)} - 3 \end{aligned}$$

Finally, we have

$$w + 3 \geq \lfloor w \rfloor + 3 \geq d(P_h) - \sum_{i=1}^{3} \sum_{j=1}^{n} e_{jh}^{(i)}.$$

□

Furthermore, the König's Theorem 2.2 verifies, that a timetable exists within the time equal to the maximal degree of the corresponding bipartite graph. In our case we can say,

1. Due to Lemmas 5.1 and 5.4, we know that the time (degree) increases maximally by 2 time units (edges) for any processor or any operation in $\mathcal{A}$.

2. Due to Lemmas 5.2 and 5.5, we know that the time (degree) increases maximally by 8 time units (edges) for any multiprocessor or any group operation in (r).

3. Due to Lemmas 5.3 and 5.6, we know that the time (degree) increases maximally by 9 time units (edges) for any processor or any individual operation in (w).

Hence we know the total time $t^{RN} \leq t_{\mathcal{A}} + 2 + \lceil r \rceil + 8 + \lceil w \rceil + 9$ and $t = t_{\mathcal{A}} + r + w$, so we have $t^{RN} \leq t + 21$. □

Remark 5.1. *In the special case of the professor-lecturer model, described by Asratian and de Werra [1], we can find a similar constant. We discuss it in the Section 5.5.*

5.4 Existence of a constant for any fixed number of multiprocessors

In this section we show that we can find for any number of p multiprocessors a constant, that gives us an upper bound for the makespan t^* with respect to the fractional makespan t.

Theorem 5.3. *The solution obtained by a rounding network, denote it by t^{RN}, for an open shop with integer preemptions and p multiprocessors is not longer then $2p \cdot (2^{p-1} - 1) + 3$ time units then the optimal makespan t for the fractional model,*

$$t^{RN} - t \leq 2p \cdot (2^{p-1} - 1) + 3.$$

Proof. The proof for any $p \geq 3$ is similar to the proof of Theorem 5.2. Here, we will have p multiprocessors and p sets of the remaining processors performing individual operations. In the body of the (RN), see Figures 5.4-5.6, where we have a splitting in 6 parallel flows in the case of $p = 3$, we have for any p, $2^p - 2$ parallel flows. But the use of the network and the roundings is exactly the same.

If we increase the rounding network in the way described above, a feasible flow is given by the solution of the LP. Furthermore, it exists a maximal integer flow by the integral flow Theorem 2.5.

We do not change the notation. We call the set of the intervals (C_i) $\forall i = 1, \ldots, k_p$ still $\mathcal{A}$ and the makespan $t_{\mathcal{A}}$ for the fractional and $t^*_{\mathcal{A}}$ for the integer case. Here, we also use the notation of $t_{\mathcal{A},J}$ and $t_{\mathcal{A},P}$ to differentiate between the bounds on the jobs and the processors. Then $e^{(i)}_{jh}$ is the flow in the arcs (P^i_h, J^*_j) and in the same way we have $d^{(i)}_{jl}$ for the arcs (g^i_l, J^*_j).

First, we show the proof for each job J_j.

Lemma 5.7. $\forall j$, *we have*

$$t^*_{\mathcal{A}} \leq t_{\mathcal{A},J} + 1.$$

Proof. Due to the bound γ (5.6), we have

$$\begin{aligned}
\sum_{i=1}^{k_p}\sum_{h=1}^{m} e^{(i)}_{jh} + \sum_{i=1}^{k_p}\sum_{l=1}^{p} d^{(i)}_{jl} &\leq \lfloor \sum_{i=1}^{k_p}\sum_{h=1}^{m} y^{(i)}_{jh} + \sum_{i=1}^{k_p}\sum_{l=1}^{p} x^{(i)}_{jl} \rfloor + 1 \\
&\leq \sum_{i=1}^{k_p}\sum_{h=1}^{m} y^{(i)}_{jh} + \sum_{i=1}^{k_p}\sum_{l=1}^{p} x^{(i)}_{jl} + 1 \\
t^*_{\mathcal{A}} &\leq t_{\mathcal{A},J} + 1
\end{aligned}$$

□

Lemma 5.8. $\forall j$, *we have*

$$\sum_{j=1}^{n} a_{jl} - \sum_{i=1}^{k_p}\sum_{l=1}^{p} d^{(i)}_{jl} \leq \lfloor r \rfloor + p \cdot k_p = \lceil r \rceil + p \cdot (2^{p-1} - 1) - 1.$$

Proof. If we define $x^{(i)}_{jl} = \lfloor x^{(i)}_{jl} \rfloor + \epsilon^{(i)}_{jl},\ 0 \leq \epsilon^{(i)}_{jl} < 1,\ \forall j, l, i$ then by the LP we know

$$\begin{aligned}
r \geq \sum_{j=1}^{n} a_{jl} - \sum_{i=1}^{k_p}\sum_{l=1}^{p} x^{(i)}_{jl} &= \sum_{j=1}^{n} a_{jl} - \sum_{i=1}^{k_p}\sum_{l=1}^{p} \lfloor x^{(i)}_{jl} \rfloor - \sum_{i=1}^{k_p}\sum_{l=1}^{p} \epsilon^{(i)}_{jl} \\
&\geq \sum_{j=1}^{n} a_{jl} - \sum_{i=1}^{k_p}\sum_{l=1}^{p} d^{(i)}_{jl} - \sum_{i=1}^{k_p}\sum_{l=1}^{p} \epsilon^{(i)}_{jl}
\end{aligned}$$

The leftover of group operations to do in (r) is then

$$\begin{aligned}
\sum_{j=1}^{n} a_{jl} - \sum_{i=1}^{k_p}\sum_{l=1}^{p} d^{(i)}_{jl} &\leq \lceil r \rceil + \lfloor \sum_{i=1}^{k_p}\sum_{l=1}^{p} \epsilon^{(i)}_{jl} \rfloor \\
&\leq \lceil r \rceil + p \cdot k_p - 1 = \lceil r \rceil + p \cdot k_p - 1 = \lceil r \rceil + p \cdot (2^{p-1} - 1) - 1.
\end{aligned}$$

□

Lemma 5.9. $\forall j$, *we have*

$$d(J_j) - \sum_{i=1}^{k_p}\sum_{h=1}^{m} e^{(i)}_{jh} \leq \lceil w \rceil + p \cdot k_p = \lceil w \rceil + p \cdot (2^{p-1} - 1).$$

Proof. If we take the entire flow in $\mathcal{A}$, then we know $\forall j$ due to the network's bound γ that

$$\begin{aligned}\sum_{i=1}^{k_p}\sum_{h=1}^{m} e_{jh}^{(i)} + \sum_{i=1}^{k_p}\sum_{l=1}^{p} d_{jl}^{(i)} &\geq \lfloor \sum_{i=1}^{k_p}\sum_{h=1}^{m} y_{jh}^{(i)} + \sum_{i=1}^{k_p}\sum_{l=1}^{p} x_{jl}^{(i)} \rfloor \\ &\geq \lfloor \sum_{i=1}^{k_p}\sum_{h=1}^{m} y_{jh}^{(i)} \rfloor + \lfloor \sum_{i=1}^{k_p}\sum_{l=1}^{p} x_{jl}^{(i)} \rfloor\end{aligned}$$

If we use the similar properties as the equations, (5.7)-(5.9), for three multiprocessors, then we have

$$\begin{aligned}\sum_{i=1}^{k_p}\sum_{h=1}^{m} e_{jh}^{(i)} + \sum_{i=1}^{k_p}\sum_{l=1}^{p} d_{jl}^{(i)} &\geq \lfloor \sum_{i=1}^{k_p}\sum_{h=1}^{m} y_{jh}^{(i)} \rfloor + \lfloor \sum_{i=1}^{k_p}\sum_{l=1}^{p} x_{jl}^{(i)} \rfloor \\ &\geq d(J_j) - \lceil w \rceil + \lfloor \sum_{i=1}^{k_p}\sum_{l=1}^{p} x_{jl}^{(i)} \rfloor \\ &\geq d(J_j) - \lceil w \rceil + \sum_{i=1}^{k_p}\sum_{l=1}^{p} \lceil x_{jl}^{(i)} \rceil - \lceil \sum_{i=1}^{k_p}\sum_{l=1}^{p} \overline{\epsilon}_{jl}^{(i)} \rceil \\ &\geq d(J_j) - \lceil w \rceil + \sum_{i=1}^{k_p}\sum_{l=1}^{p} d_{jl}^{(i)} - \lceil \sum_{i=1}^{k_p}\sum_{l=1}^{p} \overline{\epsilon}_{jl}^{(i)} \rceil \\ \lceil w \rceil + \lceil \sum_{i=1}^{k_p}\sum_{l=1}^{p} \overline{\epsilon}_{jl}^{(i)} \rceil &\geq d(J_j) - \sum_{i=1}^{k_p}\sum_{h=1}^{m} e_{jh}^{(i)} \\ \lceil w \rceil + p \cdot k_p &\geq d(J_j) - \sum_{i=1}^{k_p}\sum_{h=1}^{m} e_{jh}^{(i)}\end{aligned}$$

□

Lemma 5.10. *For every processor, we have*

$$t_{\mathcal{A}}^{*} \leq t_{\mathcal{A},P} + 2$$

.

Proof. Due to μ (5.1) for the multiprocessors and λ (5.2) for the individual proces-

sors, we have $\forall h, l$

$$\begin{aligned} t^*_{\mathcal{A}} = \sum_{i=1}^{k_p}\sum_{j=1}^{n} e^{(i)}_{jh} + \sum_{i=1}^{k_p}\sum_{j=1}^{n} d^{(i)}_{jl} &\leq \lceil \sum_{i=1}^{k_p}\sum_{j=1}^{n} y^{(i)}_{jh} \rceil + \lceil \sum_{i=1}^{k_p}\sum_{j=1}^{n} x^{(i)}_{jl} \rceil \\ &\leq \lfloor \sum_{i=1}^{k_p}\sum_{j=1}^{n} y^{(i)}_{jh} \rfloor + \lfloor \sum_{i=1}^{k_p}\sum_{j=1}^{n} x^{(i)}_{jl} \rfloor + 2 \\ &\leq \sum_{i=1}^{k_p}\sum_{j=1}^{n} y^{(i)}_{jh} + \sum_{i=1}^{k_p}\sum_{j=1}^{n} x^{(i)}_{jl} + 2 \\ &\leq t_{\mathcal{A},P} + 2 \end{aligned}$$

□

Lemma 5.11. *$\forall G_l$, we have*

$$\sum_{j=1}^{n} a_{jl} - \sum_{i=1}^{k_p}\sum_{j=1}^{n} d^{(i)}_{jl} \leq \lfloor r \rfloor + k_p = \lfloor r \rfloor + 2^{p-1} - 1.$$

Proof. Due to the bounds α (5.3), we have

$$\lfloor \sum_{j=1}^{n} x^{(i)}_{jl} \rfloor \leq \sum_{j=1}^{n} d^{(i)}_{jl} \leq \lceil \sum_{j=1}^{n} x^{(i)}_{jl} \rceil.$$

Furthermore, we can show that

$$\sum_{i=1}^{k_p}\sum_{j=1}^{n} x^{(i)}_{jl} \leq \sum_{i=1}^{k_p} \lceil \sum_{j=1}^{n} x^{(i)}_{jl} \rceil \leq \sum_{i=1}^{k_p} \lfloor \sum_{j=1}^{n} x^{(i)}_{jl} \rfloor + k_p$$

With these two properties and the LP, we show

$$\begin{aligned} r = \sum_{j=1}^{n} a_{jl} - \sum_{i=1}^{k_p}\sum_{j=1}^{n} x^{(i)}_{jl} &\geq \sum_{j=1}^{n} a_{jl} - \sum_{i=1}^{k_p} \lfloor \sum_{j=1}^{n} x^{(i)}_{jl} \rfloor - k_p \\ &\geq \sum_{j=1}^{n} a_{jl} - \sum_{i=1}^{k_p}\sum_{j=1}^{n} d^{(i)}_{jl} - k_p \end{aligned}$$

Finally we obtain

$$r + k_p \geq \lfloor r \rfloor + k_p \geq \sum_{j=1}^{n} a_{jl} - \sum_{i=1}^{k_p}\sum_{j=1}^{n} d^{(i)}_{jl}.$$

□

Lemma 5.12. *$\forall P_h$, we have*

$$d(P_h) - \sum_{i=1}^{k_p}\sum_{j=1}^{n} e_{jh}^{(i)} \leq \lfloor w \rfloor + k_p = \lfloor w \rfloor + 2^{p-1} - 1.$$

Proof. Due to the bounds β (5.4), we have

$$\lfloor \sum_{j=1}^{n} y_{jh}^{(i)} \rfloor \leq \sum_{j=1}^{n} e_{jh}^{(i)} \leq \lceil \sum_{j=1}^{n} y_{jh}^{(i)} \rceil$$

Furthermore, we know that

$$\sum_{i=1}^{k_p}\sum_{j=1}^{n} y_{jh}^{(i)} \leq \sum_{i=1}^{k_p} \lceil \sum_{j=1}^{n} y_{jh}^{(i)} \rceil \leq \sum_{i=1}^{k_p} \lfloor \sum_{j=1}^{n} y_{jh}^{(i)} \rfloor + k_p$$

Then we can show,

$$\begin{aligned} w \geq d(P_h) - \sum_{i=1}^{k_p}\sum_{j=1}^{n} y_{jh}^{(i)} &\geq d(P_h) - \sum_{i=1}^{k_p} \lfloor \sum_{j=1}^{n} y_{jh}^{(i)} \rfloor - k_p \\ &\geq d(P_h) - \sum_{i=1}^{k_p}\sum_{j=1}^{n} e_{jh}^{(i)} - k_p \end{aligned}$$

Finally we obtain

$$w + k_p \geq \lfloor w \rfloor + k_p \geq d(P_h) - \sum_{i=1}^{k_p}\sum_{j=1}^{n} e_{jh}^{(i)}.$$

□

Moreover, due to the König's Theorem 2.2, a timetable exists within the time equal to the maximal degree of the corresponding bipartite graph. In our case we have,

1. Due to Lemmas 5.7 and 5.10, we know that the time (degree) increases maximally by 2 time units (edges) for any processor or any operation in $\mathcal{A}$.

2. Due to Lemmas 5.8 and 5.11, we know that the time (degree) increases maximally by $p \cdot (2^{p-1} - 1) - 1$ time units (edges) for any multiprocessor or any group operation in (r).

3. Due to Lemmas 5.9 and 5.12, we know that the time (degree) increases maximally by $p \cdot (2^{p-1} - 1)$ time units (edges) for any processor or any individual operation in (w).

Hence we know the total time $t^{RN} \leq t_{\mathcal{A}} + 2 + \lceil r \rceil + p \cdot (2^{p-1} - 1) - 1 + \lceil w \rceil + p \cdot (2^{p-1} - 1)$, so we have $t^{RN} \leq t + 2p \cdot (2^{p-1} - 1) + 3$.

□

5.5 A constant error for the professor-lecturer model

The professor-lecturer model was introduced by Asratian and de Werra [1]. Here we demonstrate an approximation algorithm with a constant absolute error for a fixed number of multiprocessors. First, we review the model and give a more appropriate introduction.

5.5.1 The professor-lecturer model

As introduced in Chapter 3, the main difference between the professor-lecturer model and the general open shop is that in the professor-lecturer model a group operation is given by a professor and individual operations are done by lecturers. In other words a professor gives only group lectures and on the other side a lecturer gives only individual lectures to single classes. Classes in the professor-lecturer model are processors in the general open shop, and the teachers are the jobs.

For this model, Asratian (see [1]) has shown to obtain a timetable with makespan t^*, provided that each teacher, professor or lecturer, has at most $\frac{t^*}{2}$ lectures and each class has not more then t^* lectures and not more then $\frac{t^*}{2}$ group lectures. Moreover, Asratian and de Werra in [1] have shown following theorem.

Theorem 5.4. *In the case of the professor-lecturer model, where*

$$W_1 = \max_{l,h}(\sum_{j=1}^{n} a_{jl} + \sum_{h=1}^{m} b_{jh}),$$

we have

$$W \leq t^* \leq \max(W, W_1).$$

The proof is given in [1].

To determine the deviation between the optimal makespan with fractional preemptions and the makespan with integer preemptions, we pursue the same procedure as in Section 5.4. The difference is hidden behind the bounds for the jobs, because there is the split between individual and group lectures.

Remark 5.2. *In the professor-lecturer model we have m classes and n jobs. But because of the difference between professors and lecturers, we have p professors and $n-p$ lecturers. So for the formulation with three professors, we have J_1, J_2, J_3 corresponding to the three professors. In the same way, $J_4, \ldots, J_n$ correspond to the lecturers.*

Moreover we assume each professor has at least one lecture, otherwise we can reduce p.

5.5.2 The LP formulation for the professor-lecturer model with 3 professors

Because of the split for the jobs in professors and lecturers, the LP formulation (4.1) becomes smaller because we need fewer constraints. We give the LP formulation for three professors below. For more then three professors the construction for the LP follows in a similar manner as for $p = 3$. The LP depends on the structure given in Figure 4.5.

$$\min(w - 2r - a_1 - b_1 - c_1) \tag{5.14}$$

$$\begin{aligned}
0 \leq \sum_{i=1}^{3} y_{jh}^{(i)} &\leq b_{jh},\ \forall j = 4, \ldots, n,\ \forall h = 1, \ldots, m \\
0 \leq \sum_{i=1}^{3} x_{jl}^{(i)} &\leq a_{jl},\ \forall j = 1, 2, 3,\ l = 1, 2, 3 \\
\sum_{h=4}^{m} y_{jh}^{(1)} &\leq a_1,\ \forall j = 4, \ldots, n \\
x_{j1}^{(1)} + x_{j2}^{(1)} &\leq a_1,\ \forall j = 1, 2, 3 \\
\sum_{h=1}^{m} y_{jh}^{(2)} &\leq b_1,\ \forall j = 4, \ldots, n \\
x_{j1}^{(2)} + x_{j3}^{(2)} &\leq b_1,\ \forall j = 1, 2, 3 \\
\sum_{h=1}^{m} y_{jh}^{(3)} &\leq c_1,\ \forall j = 4, \ldots, n \\
x_{j2}^{(3)} + x_{j3}^{(3)} &\leq c_1,\ \forall j = 1, 2, 3 \\
\sum_{j=4}^{n} y_{jh}^{(1)} &\leq a_1,\ \forall h = 1, \ldots, m \\
\sum_{j=4}^{n} y_{jh}^{(2)} &\leq b_1,\ \forall h = 1, \ldots, m \\
\sum_{j=4}^{n} y_{jh}^{(3)} &\leq c_1,\ \forall h = 1, \ldots, m \\
\sum_{j=1}^{3} x_{jl}^{(1)} &= a_1,\ l = 1, 2, 3
\end{aligned}$$

$$
\begin{aligned}
\sum_{j=1}^{3} x_{jl}^{(2)} &= b_1,\ l = 1,2,3 \\
\sum_{j=1}^{3} x_{jl}^{(3)} &= c_1,\ l = 1,2,3 \\
\sum_{h=1}^{m} y_{jh}^{(1)} &\leq a_2,\ \forall j = 4,\ldots,n \\
x_{j1}^{(1)} + x_{j3}^{(1)} &\leq a_2,\ \forall j = 1,2,3 \\
\sum_{h=1}^{m} y_{jh}^{(2)} &\leq b_2,\ \forall j = 4,\ldots,n \\
x_{j2}^{(2)} &\leq b_2,\ \forall j = 1,2,3 \\
\sum_{h=1}^{m} y_{jh}^{(3)} &\leq c_2,\ \forall j = 4,\ldots,n \\
x_{j1}^{(3)} &\leq c_2,\ \forall j = 1,2,3 \\
\sum_{j=4}^{n} y_{jh}^{(1)} &\leq a_2,\ \forall h = 1,\ldots,m \\
\sum_{j=4}^{n} y_{jh}^{(2)} &\leq b_2,\ \forall h = 1,\ldots,m \\
\sum_{j=4}^{n} y_{jh}^{(3)} &\leq c_2,\ \forall h = 1,\ldots,m \\
\sum_{j=1}^{3} x_{j3}^{(1)} &= a_2 \\
\sum_{j=1}^{3} x_{j2}^{(2)} &= b_2 \\
\sum_{j=1}^{3} x_{j1}^{(3)} &= c_2 \\
\sum_{l=1}^{3} a_{jl} - \sum_{l=1}^{3}\sum_{i=1}^{3} x_{jl}^{(i)} &\leq r,\ \forall j = 1,2,3 \qquad (5.15) \\
\sum_{j=1}^{3} a_{jl} - \sum_{j=1}^{3}\sum_{i=1}^{3} x_{jl}^{(i)} &= r,\ l = 1,2,3 \qquad (5.16) \\
d(J_j) - \sum_{h=1}^{m}\sum_{i=1}^{3} y_{jh}^{(i)} &\leq w,\ \forall j = 4,\ldots,n \qquad (5.17) \\
d(P_h) - \sum_{j=4}^{n}\sum_{i=1}^{3} y_{jh}^{(i)} &\leq w,\ \forall h = 1,\ldots,m \qquad (5.18)
\end{aligned}
$$

5.5.3 The rounding network (RN) for the professor-lecturer model

The form of the rounding network (RN) is quite similar to the general open shop. The difference due to the job partition appears in the bounds and in some arcs. All arcs, where a multiprocessor is joined with a job J_j with $j \geq 4$ or a individual processor is joined with a J_j with $j = 1, \ldots, 3$, are unnecessary and are disappearing.

If we start with the bounds for μ and λ, then we have to lower the range dependent on j.

$$\mu_l = (\lfloor \sum_{i=1}^{3} \sum_{j=1}^{3} x_{jl}^{(i)} \rfloor, \lceil \sum_{i=1}^{3} \sum_{j=1}^{3} x_{jl}^{(i)} \rceil), \ \forall l = 1, 2, 3; \tag{5.19}$$

$$\lambda_h = (\lfloor \sum_{i=1}^{3} \sum_{j=4}^{n} y_{jh}^{(i)} \rfloor, \lceil \sum_{i=1}^{3} \sum_{j=4}^{n} y_{jh}^{(i)} \rceil), \ \forall h = 1, \ldots, m; \tag{5.20}$$

The bounds for α and β also change only with respect to the index j, in exactly the same way as μ and λ.

$$\alpha_l^i = (\lfloor \sum_{j=1}^{3} x_{jl}^{(i)} \rfloor, \lceil \sum_{j=1}^{3} x_{jl}^{(i)} \rceil), \ \forall l = 1, 2, 3, \ i = 1, 2, 3; \tag{5.21}$$

$$\beta_h^i = (\lfloor \sum_{j=4}^{n} y_{jh}^{(i)} \rfloor, \lceil \sum_{j=4}^{n} y_{jh}^{(i)} \rceil), \ \forall h = 1, \ldots, m, \ i = 1, 2, 3; \tag{5.22}$$

The bounds in (RN) represented by θ and η do not change, but we have less arcs because we do not have any correspondence between multiprocessors and any J_j with $j \geq 4$. Conversely we have never a individual processor joined with J_j where $j = 1, \ldots, 3$. Hence we have

$$\begin{aligned} \theta_{jl}^i &= (\lfloor x_{jl}^{(i)} \rfloor, \lceil x_{jl}^{(i)} \rceil), \ \forall j = 1, \ldots, 3, \ l = 1, 2, 3, \ i = 1, 2, 3; \\ \eta_{jh}^i &= (\lfloor y_{jh}^{(i)} \rfloor, \lceil y_{jh}^{(i)} \rceil), \ \forall j = 4, \ldots, n, \ h = 1, \ldots, m, \ i = 1, 2, 3; \end{aligned} \tag{5.23}$$

For σ we present only the changes on σ_1^1, all the others are similar.

$$\begin{aligned} \sigma_1^1 &= (\lfloor x_{j1}^{(1)} + x_{12}^{(1)} \rfloor, \lceil x_{11}^{(1)} + x_{12}^{(1)} \rceil), \ \forall j = 1, 2, 3; \\ \sigma_1^1 &= (\lfloor y_{1h_3}^{(1)} \rfloor, \lceil y_{1h_3}^{(1)} \rceil), \ \forall j = 4, \ldots, n; \end{aligned}$$

And with respect to

$$\begin{aligned}\gamma_j &= (\lfloor\sum_{i=1}^{3}\sum_{l=1}^{3} x_{jl}^{(i)}\rfloor, \lceil\sum_{i=1}^{3}\sum_{l=1}^{3} x_{jl}^{(i)}\rceil),\ \forall j = 1,2,3;\\ \gamma_j &= (\lfloor\sum_{i=1}^{3}\sum_{h=1}^{m} y_{jh}^{(i)}\rfloor, \lceil\sum_{i=1}^{3}\sum_{h=1}^{m} y_{jh}^{(i)}\rceil),\ \forall j = 4,\ldots,n;\end{aligned} \tag{5.24}$$

5.5.4 A constant for the professor-lecturer model with 3 multiprocessors

The existence of a constant deviation between the optimal makespan for the fractional model and the professor-lecturer model is given by the proof for the open shop. Thus we know that the professor-lecturer model is a simplification of the jobs.

Theorem 5.5. *The solution obtained by a rounding network, denote it by t^{RN}, for the professor-lecturer model with three multiprocessors is not longer then 14 time units then the optimal makespan t for the fractional model,*

$$t^{RN} - t \leq 14.$$

Proof. The existence of a feasible flow is given by the LP solution of the LP (5.14). Moreover, we know that it exists as a maximal integer flow by the integral flow Theorem 2.5.

Furthermore, we show the corresponding lemmas that assure we do not exceed the leftover in the precedent proof. There we have three corresponding to the jobs and three for processors: one on $\mathcal{A}$, one on the overlap (r) and one on (w).

Lemma 5.13. *$\forall j$ we have $t^*_{\mathcal{A}} \leq t_{\mathcal{A},J} + 1$.*

Proof. Due to the network's bound γ, see equation (5.24), we have for $j = 4,\ldots,n$

$$\sum_{i=1}^{3}\sum_{h=1}^{m} e_{jh}^{(i)} \leq \lceil\sum_{i=1}^{3}\sum_{h=1}^{m} y_{jh}^{(i)}\rceil$$

Then for $j = 1,2,3$ we have

$$\sum_{i=1}^{3}\sum_{l=1}^{3} d_{jl}^{(i)} \leq \lceil\sum_{i=1}^{3}\sum_{l=1}^{3} x_{jl}^{(i)}\rceil$$

Therefore we know $\forall j$ we have

$$t^*_{\mathcal{A}} \leq t_{\mathcal{A},J} + 1.$$

□

Lemma 5.14. $\forall j \qquad \sum_{j=1}^{3} a_{jl} - \sum_{i=1}^{3}\sum_{l=1}^{3} d_{jl}^{(i)} \leq \lceil r \rceil + 8.$

Proof. If we define $x_{jl}^{(i)} = \lfloor x_{jl}^{(i)} \rfloor + \epsilon_{jl}^{(i)},\ 0 \leq \epsilon_{jl}^{(i)} < 1,\ \forall j,l,i$ and with the LP equation (5.15), we know

$$\begin{aligned} r \geq \sum_{j=1}^{3} a_{jl} - \sum_{i=1}^{3}\sum_{l=1}^{3} x_{jl}^{(i)} &= \sum_{j=1}^{3} a_{jl} - \sum_{i=1}^{3}\sum_{l=1}^{3} \lfloor x_{jl}^{(i)} \rfloor - \sum_{i=1}^{3}\sum_{l=1}^{3} \epsilon_{jl}^{(i)} \\ &\geq \sum_{j=1}^{3} a_{jl} - \sum_{i=1}^{3}\sum_{l=1}^{3} d_{jl}^{(i)} - \sum_{i=1}^{3}\sum_{l=1}^{3} \epsilon_{jl}^{(i)} \end{aligned}$$

The leftover of group operations to do in (r) is then

$$\begin{aligned} \sum_{j=1}^{3} a_{jl} - \sum_{i=1}^{3}\sum_{l=1}^{3} d_{jl}^{(i)} &\leq \lfloor r + \sum_{i=1}^{3}\sum_{l=1}^{3} \epsilon_{jl}^{(i)} \rfloor \\ &\leq \lceil r \rceil + \lfloor \sum_{i=1}^{3}\sum_{l=1}^{3} \epsilon_{jl}^{(i)} \rfloor \\ &\leq \lceil r \rceil + 8. \end{aligned}$$

□

Lemma 5.15. $\forall j = 4, \ldots, n \qquad d(J_j) - \sum_{i=1}^{3}\sum_{h=1}^{m} e_{jh}^{(i)} \leq \lceil w \rceil.$

Proof. Due to the γ bound (5.24),we have

$$\sum_{i=1}^{3}\sum_{h=1}^{m} e_{jh}^{(i)} \geq \lfloor \sum_{i=1}^{3}\sum_{h=1}^{m} y_{jh}^{(i)} \rfloor.$$

So we can say with the help of equation (5.17)

$$\sum_{i=1}^{3}\sum_{h=1}^{m} e_{jh}^{(i)} \geq d(J_j) - \lceil w \rceil.$$

So we obtain

$$d(J_j) - \sum_{i=1}^{3}\sum_{h=1}^{m} e_{jh}^{(i)} \leq \lceil w \rceil.$$

□

Now we have to prove that we can bound the leftover with respect to each multiprocessor and each individual processor.

Lemma 5.16. *For every processor we have*

$$t^*_{\mathcal{A}} \leq t_{\mathcal{A},P} + 1$$

.

Proof. Due to the bounds μ (5.19) and λ (5.20) in (RN), we have $\forall h, l$

$$\sum_{i=1}^{3}\sum_{j=1}^{3} d^{(i)}_{jl} \leq \lceil \sum_{i=1}^{3}\sum_{j=1}^{3} x^{(i)}_{jl} \rceil,$$

$$\sum_{i=1}^{3}\sum_{j=4}^{n} e^{(i)}_{jh} \leq \lceil \sum_{i=1}^{3}\sum_{j=4}^{n} y^{(i)}_{jh} \rceil.$$

There we obtain

$$t^*_{\mathcal{A}} \leq t_{\mathcal{A},P} + 1.$$

□

Lemma 5.17. *For each multiprocessor G_l, $l = 1, 2$ or 3, we have*

$$\sum_{j=1}^{3} a_{jl} - \sum_{i=1}^{3}\sum_{j=1}^{3} d^{(i)}_{jl} \leq \lfloor r \rfloor + 3.$$

Proof. Due to the bounds α (5.21) we have

$$\lfloor \sum_{j=1}^{3} x^{(i)}_{jl} \rfloor \leq \sum_{j=1}^{n} d^{(i)}_{jl} \leq \lceil \sum_{j=1}^{3} x^{(i)}_{jl} \rceil \tag{5.25}$$

Furthermore, we can show that

$$\sum_{i=1}^{3}\sum_{j=1}^{3} x^{(i)}_{jl} \leq \sum_{i=1}^{3} \lceil \sum_{j=1}^{3} x^{(i)}_{jl} \rceil \leq \sum_{i=1}^{3} \lfloor \sum_{j=1}^{3} x^{(i)}_{jl} \rfloor + 3 \tag{5.26}$$

Now with these equations, we show

$$\begin{aligned} r = \sum_{j=1}^{3} a_{jl} - \sum_{i=1}^{3}\sum_{j=1}^{n} x^{(i)}_{jl} &\geq \sum_{j=1}^{3} a_{jl} - \sum_{i=1}^{3} \lceil \sum_{j=1}^{3} x^{(i)}_{jl} \rceil \\ &\geq \sum_{j=1}^{3} a_{jl} - \sum_{i=1}^{3} \lfloor \sum_{j=1}^{3} x^{(i)}_{jl} \rfloor - 3 \\ &\geq \sum_{j=1}^{3} a_{jl} - \sum_{i=1}^{3}\sum_{j=1}^{3} d^{(i)}_{jl} - 3 \end{aligned}$$

Finally we obtain

$$r + 3 \geq \lfloor r \rfloor + 3 \geq \sum_{j=1}^{3} a_{jl} - \sum_{i=1}^{3} \sum_{j=1}^{3} d_{jl}^{(i)}.$$

□

Lemma 5.18. *For each processor P_h, $4 \leq h \leq m$, we have*

$$d(P_h) - \sum_{i=1}^{3} \sum_{j=4}^{n} e_{jh}^{(i)} \leq \lfloor w \rfloor + 3.$$

Proof. Due to the bounds β (5.22), we have

$$\lfloor \sum_{j=4}^{n} y_{jh}^{(i)} \rfloor \leq \sum_{j=1}^{n} e_{jh}^{(i)} \leq \lceil \sum_{j=4}^{n} y_{jh}^{(i)} \rceil \tag{5.27}$$

Then we know that

$$\sum_{i=1}^{3} \sum_{j=4}^{n} y_{jh}^{(i)} \leq \sum_{i=1}^{3} \lceil \sum_{j=4}^{n} y_{jh}^{(i)} \rceil \leq \sum_{i=1}^{3} \lfloor \sum_{j=4}^{n} y_{jh}^{(i)} \rfloor + 3 \tag{5.28}$$

Now we show, including the equations (5.27), (5.28) and the LP equation (5.18),

$$\begin{aligned} w \geq d(P_h) - \sum_{i=1}^{3} \sum_{j=4}^{n} y_{jh}^{(i)} &\geq d(P_h) - \sum_{i=1}^{3} \lceil \sum_{j=4}^{n} y_{jh}^{(i)} \rceil \\ &\geq d(P_h) - \sum_{i=1}^{3} \lfloor \sum_{j=4}^{n} y_{jh}^{(i)} \rfloor - 3 \\ &\geq d(P_h) - \sum_{i=1}^{3} \sum_{j=4}^{n} e_{jh}^{(i)} - 3 \end{aligned}$$

Finally we have

$$w + 3 \geq \lfloor w \rfloor + 3 \geq d(P_h) - \sum_{i=1}^{3} \sum_{j=4}^{n} e_{jh}^{(i)}.$$

□

Furthermore, the König's Theorem 2.2 verifies, that it exists as a timetable within the time equal to the maximal degree of the corresponding bipartite graph. In our case we can have:

1. From Lemmas 5.13 and 5.16, we know that the time (degree) increases maximally by 1 time unit (edge) for any processor or any operation in $\mathcal{A}$.

2. From Lemmas 5.14 and 5.17, we know that the time (degree) increases maximally by 8 time units (edges) for any multiprocessor or any group operation in (r).

3. From Lemmas 5.15 and 5.18, we know that the time (degree) increases maximally by 3 time units (edges) for any processor or any individual operation in (w).

Hence we know the total time $t^{RN} \leq t_{\mathcal{A}} + 1 + \lceil r \rceil + 8 + \lceil w \rceil + 3$, so we have $t^{RN} \leq t + 12$. □

5.5.5 A constant for any p in the professor-lecturer model

As in the general open shop we also tried to find the deviation for any p of multiprocessors, or as they are called in the professor-lecturer model, professors. Therefore we have following theorem.

Theorem 5.6. *The solution obtained by a rounding network, denote it by t^{RN}, for the professor-lecturer model with p multiprocessors is not longer then $(p+1)(2^{p-1}-1)+2$ time units then the optimal makespan t for the fractional model,*

$$t^{RN} - t \leq (p+1)(2^{p-1}-1) + 2.$$

Proof. The proof for this case is equivalent to the extension for the general open shop. We take the network for the professor-lecturer model and increase the bounds with respect to the index l, goes now form 1 to p, and index i, goes from 1 to k_p. The value of k_p is still $2^{p-1} - 1$.

We give no further explanations, because it has exactly the same structure as the model with three professors. So we proof directly the six lemmas.

Lemma 5.19. *$\forall j$ we have $t^*_{\mathcal{A}} \leq t_{\mathcal{A},J} + 1$.*

Proof. Due to the network's bound γ, we have for $j = 4, \ldots, n$

$$\sum_{i=1}^{k_p} \sum_{h=1}^{m} e_{jh}^{(i)} \leq \lceil \sum_{i=1}^{k_p} \sum_{h=1}^{m} y_{jh}^{(i)} \rceil$$

Then for $j = 1, 2, 3$ we have

$$\sum_{i=1}^{k_p} \sum_{l=1}^{p} d_{jl}^{(i)} \leq \lceil \sum_{i=1}^{k_p} \sum_{l=1}^{p} x_{jl}^{(i)} \rceil$$

Therefore we know $\forall j$ we have

$$t^*_{\mathcal{A}} \leq t_{\mathcal{A},J} + 1.$$

□

Lemma 5.20. $\forall j \qquad \sum_{j=1}^{3} a_{jl} - \sum_{i=1}^{k_p}\sum_{l=1}^{p} d_{jl}^{(i)} \leq \lceil r \rceil + p \cdot (2^{p-1} - 1) - 1.$

Proof. If we define $x_{jl}^{(i)} = \lfloor x_{jl}^{(i)} \rfloor + \epsilon_{jl}^{(i)},\ 0 \leq \epsilon_{jl}^{(i)} < 1,\ \forall j, l, i$ and with generalization of the LP equation (5.15), we know

$$\begin{aligned} r \geq \sum_{j=1}^{3} a_{jl} - \sum_{i=1}^{k_p}\sum_{l=1}^{p} x_{jl}^{(i)} &= \sum_{j=1}^{3} a_{jl} - \sum_{i=1}^{k_p}\sum_{l=1}^{p} \lfloor x_{jl}^{(i)} \rfloor - \sum_{i=1}^{k_p}\sum_{l=1}^{p} \epsilon_{jl}^{(i)} \\ &\geq \sum_{j=1}^{3} a_{jl} - \sum_{i=1}^{k_p}\sum_{l=1}^{p} d_{jl}^{(i)} - \sum_{i=1}^{k_p}\sum_{l=1}^{p} \epsilon_{jl}^{(i)} \end{aligned}$$

The leftover of group operations to do in (r) is then

$$\begin{aligned} \sum_{j=1}^{3} a_{jl} - \sum_{i=1}^{k_p}\sum_{l=1}^{p} d_{jl}^{(i)} &\leq \lfloor r + \sum_{i=1}^{k_p}\sum_{l=1}^{p} \epsilon_{jl}^{(i)} \rfloor \\ &\leq \lceil r \rceil + \lfloor \sum_{i=1}^{k_p}\sum_{l=1}^{p} \epsilon_{jl}^{(i)} \rfloor \\ &\leq \lceil r \rceil + p \cdot k_p - 1 = \lceil r \rceil + p \cdot (2^{p-1} - 1) - 1. \end{aligned}$$

□

Lemma 5.21. $\forall j = 4, \ldots, n \qquad d(J_j) - \sum_{i=1}^{k_p}\sum_{h=1}^{m} e_{jh}^{(i)} \leq \lceil w \rceil.$

Proof. Due to the bound γ,we have

$$\sum_{i=1}^{k_p}\sum_{h=1}^{m} e_{jh}^{(i)} \geq \lfloor \sum_{i=1}^{k_p}\sum_{h=1}^{m} y_{jh}^{(i)} \rfloor.$$

So we can say with the help of the LP, that

$$\sum_{i=1}^{k_p}\sum_{h=1}^{m} e_{jh}^{(i)} \geq d(J_j) - \lceil w \rceil.$$

So we obtain

$$d(J_j) - \sum_{i=1}^{k_p}\sum_{h=1}^{m} e_{jh}^{(i)} \leq \lceil w \rceil.$$

□

Lemma 5.22. *For every processor we have*

$$t^*_{\mathcal{A}} \leq t_{\mathcal{A},P} + 1$$

.

Proof. Due to the bounds μ and λ in (RN), we have $\forall h, l$

$$\sum_{i=1}^{k_p}\sum_{j=1}^{3} d^{(i)}_{jl} \leq \lceil \sum_{i=1}^{k_p}\sum_{j=1}^{3} x^{(i)}_{jl} \rceil,$$

$$\sum_{i=1}^{k_p}\sum_{j=4}^{n} e^{(i)}_{jh} \leq \lceil \sum_{i=1}^{k_p}\sum_{j=4}^{n} y^{(i)}_{jh} \rceil.$$

Thus we obtain

$$t^*_{\mathcal{A}} \leq t_{\mathcal{A},P} + 1.$$

□

Lemma 5.23. *For each multiprocessor G_l, $l = 1, \ldots, p$, we have*

$$\sum_{j=1}^{3} a_{jl} - \sum_{i=1}^{k_p}\sum_{j=1}^{3} d^{(i)}_{jl} \leq \lfloor r \rfloor + 2^{p-1} - 1.$$

Proof. Due to the bounds α, we have

$$\lfloor \sum_{j=1}^{3} x^{(i)}_{jl} \rfloor \leq \sum_{j=1}^{n} d^{(i)}_{jl} \leq \lceil \sum_{j=1}^{3} x^{(i)}_{jl} \rceil \tag{5.29}$$

Furthermore, we can show that

$$\sum_{i=1}^{k_p}\sum_{j=1}^{3} x^{(i)}_{jl} \leq \sum_{i=1}^{k_p} \lceil \sum_{j=1}^{3} x^{(i)}_{jl} \rceil \leq \sum_{i=1}^{k_p} \lfloor \sum_{j=1}^{3} x^{(i)}_{jl} \rfloor + k_p \tag{5.30}$$

Now with these equations, we show

$$\begin{aligned} r = \sum_{j=1}^{3} a_{jl} - \sum_{i=1}^{k_p}\sum_{j=1}^{n} x^{(i)}_{jl} &\geq \sum_{j=1}^{3} a_{jl} - \sum_{i=1}^{k_p} \lceil \sum_{j=1}^{3} x^{(i)}_{jl} \rceil \\ &\geq \sum_{j=1}^{3} a_{jl} - \sum_{i=1}^{k_p} \lfloor \sum_{j=1}^{3} x^{(i)}_{jl} \rfloor - k_p \\ &\geq \sum_{j=1}^{3} a_{jl} - \sum_{i=1}^{k_p}\sum_{j=1}^{3} d^{(i)}_{jl} - k_p \end{aligned}$$

Finally we obtain

$$r + k_p \geq \lfloor r \rfloor + k_p \geq \sum_{j=1}^{3} a_{jl} - \sum_{i=1}^{k_p} \sum_{j=1}^{3} d_{jl}^{(i)}.$$

□

Lemma 5.24. *For each processor P_h, $4 \leq h \leq m$, we have*

$$d(P_h) - \sum_{i=1}^{k_p} \sum_{j=4}^{n} e_{jh}^{(i)} \leq \lfloor w \rfloor + 2^{p-1} - 1.$$

Proof. Due to the bounds β, we have

$$\lfloor \sum_{j=4}^{n} y_{jh}^{(i)} \rfloor \leq \sum_{j=1}^{n} e_{jh}^{(i)} \leq \lceil \sum_{j=4}^{n} y_{jh}^{(i)} \rceil \tag{5.31}$$

Then we know that

$$\sum_{i=1}^{k_p} \sum_{j=4}^{n} y_{jh}^{(i)} \leq \sum_{i=1}^{k_p} \lceil \sum_{j=4}^{n} y_{jh}^{(i)} \rceil \leq \sum_{i=1}^{k_p} \lfloor \sum_{j=4}^{n} y_{jh}^{(i)} \rfloor + k_p \tag{5.32}$$

Now we show, including the equations (5.31), (5.32) and the LP,

$$\begin{aligned} w \geq d(P_h) - \sum_{i=1}^{k_p} \sum_{j=4}^{n} y_{jh}^{(i)} &\geq d(P_h) - \sum_{i=1}^{k_p} \lceil \sum_{j=4}^{n} y_{jh}^{(i)} \rceil \\ &\geq d(P_h) - \sum_{i=1}^{k_p} \lfloor \sum_{j=4}^{n} y_{jh}^{(i)} \rfloor - k_p \\ &\geq d(P_h) - \sum_{i=1}^{k_p} \sum_{j=4}^{n} e_{jh}^{(i)} - k_p \end{aligned}$$

Finally we have

$$w + k_p \geq \lfloor w \rfloor + k_p \geq d(P_h) - \sum_{i=1}^{3} \sum_{j=4}^{n} e_{jh}^{(i)}.$$

□

Furthermore, the König's Theorem 2.2 verify, that it exists as a timetable within the time equal to the maximal degree of the corresponding bipartite graph. In our case we can have,

1. From Lemmas 5.19 and 5.22, we know that the time (degree) increases maximally by 1 time unit (edge) for any processor or any operation in $\mathcal{A}$.

2. From Lemmas 5.20 and 5.23, we know that the time (degree) increases maximally by $p \cdot (2^{p-1} - 1) - 1$ time units (edges) for any multiprocessor or any group operation in (r).

3. From Lemmas 5.21 and 5.24, we know that the time (degree) increases maximally by $2^{p-1} - 1$ time units (edges) for any processor or any individual operation in (w).

Hence, we know the total time $t^{RN} \leq t_{\mathcal{A}} + 1 + \lceil r \rceil + p \cdot (2^{p-1} - 1) - 1 + \lceil w \rceil + (2^{p-1} - 1)$, so we have $t^{RN} \leq t + (p+1)(2^{p-1} - 1) + 2$. □

Chapter 6

An approximation algorithm for (PI) using edge coloring

In this chapter we introduce a new algorithm for (PI). This algorithm works for general open shops. The timetable obtained by this algorithm needs at most

$$\max\left(\max_j w(J_j), \max_h w(P_h)\right) + \max_j \sum_{l=1}^{p} a_{jl}$$

different colors; this number is equal to the makespan of the timetable. Moreover,

$$W = \max\left(\max_j w(J_j), \max_h w(P_h)\right)$$

is called the workload and equal to the maximal degree in the graph. $w(J_j)$ is called the workload for the job J_j,

$$w(J_j) = \sum_{l=1}^{p} a_{jl} + \sum_{h=1}^{m} b_{jh}$$

and similarly $w(P_h)$ is the workload for the processor P_h,

$$w(P_h) = \sum_{j=1}^{n} a_{jl} + \sum_{j=1}^{n} b_{jh}.$$

So we can reformulate the upper bound for the makespan as

$$W + \max_j \sum_{l=1}^{p} a_{jl}.$$

A trivial upper bound is

$$\max\left(\max_j(\sum_{h=1}^{m} b_{jh}), \max_h(\sum_{j=1}^{n} b_{jh})\right) + \max\left(\max_j(\sum_{l=1}^{p} a_{jl}), \max_l(\sum_{j=1}^{n} a_{jl})\right),$$

that can be interpreted as the sum of the maximum workload restricted to individual operations and the maximum workload restricted to group operations. Or considered graph theoretical, it is the sum of colors to color the individual operations and the group operations separately. Asratian and de Werra present this algorithm in [1], we will discuss it in more detail in Chapter 7.

First we introduce a transformation, which takes the corresponding hypergraph of the open shop problem and returns a multigraph without loops. Then we present the algorithm with its optimality and proof.

Remark 6.1. *All hyperedges $|E_i| \geq 3$ of a hypergraph represent one time unit of a group operation. The hyperedges with $|E_i| = 2$ represent an individual operation and those with $|E_i| = 1$, called loops, does not exist in general open shops.*

6.1 The transformation

The transformation of a hypergraph H into a graph G occurs in the following steps:

1. All vertices of H do not change;
2. All hyperedges E_i, for which $|E_i| \leq 2$, do not change;
3. All hyperedges E_i, for which $|E_i| \geq 3$, are transformed into a besom;

(Step 3 is explained in the next subsection).

6.1.1 Transformation of a hyperedge $|E_i| \geq 3$ into a besom

Each hyperedge $|E_i| \geq 3$ is changed in such a way, that we create a new vertex for each of them. This new vertex is called an *intervertex*. Here we join each vertex included in E_i with the intervertex. We have a bipartite hypergraph with the processors on one side and the jobs on the other. All the edges between the processors and the intervertex are called *besom fingers*. The edge joining the job and the intervertex is called *besom head*. The subgraph containing all edges attached at the intervertex is called a *besom*.

Example 6.1. *Consider the hypergraph in Figure 6.1 with 4 hyperedges for which $|E_i| \geq 3$ (group operations); they are drawn as polygons. The hyperedges with $|E_i| = 2$ (individual operations) are drawn as dashed lines. Subsequent to the transformation, we obtained the graph G in Figure 6.2 where the vertices and hyperedges with $|E_i| = 2$ were not transformed. Each polygon is now substitued by an besom where the appropriate processors are joined with the intervertex and this one with the appropriate job.*

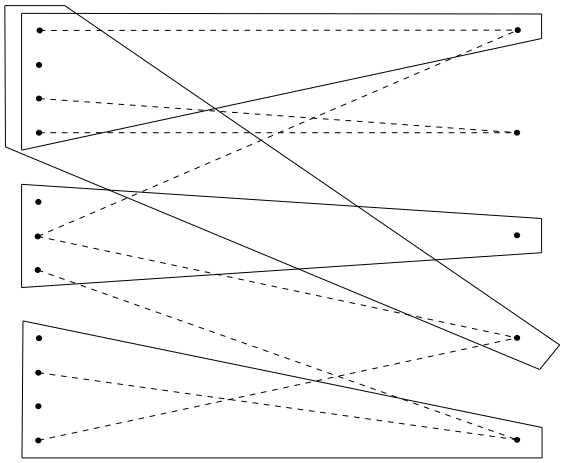

Figure 6.1: The hypergraph H.

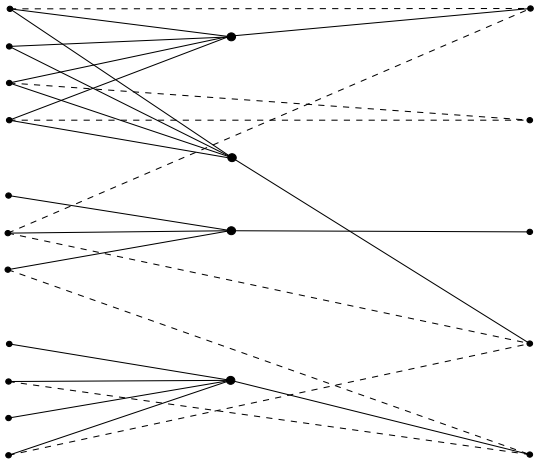

Figure 6.2: The resulting graph G.

6.1.2 Properties concerning besoms

Important for the subsequent devolution, is the special behavior of an intervertex in a besom. The intervertex has the same properties as any other vertex, but a intervertex has always degree zero by definition. Hence, if we search the vertex having maximal degree in graph G, containing besoms, the maximal degree is found on a vertex and not on a intervertex.

Propositon 6.1. *The degree for an intervertex per definition is always zero.*

Moreover, if we color the edges of a graph containing besoms, we have to color the edges in such a way, that all edges included in the besom, the besom fingers and the besom head, have the same color.

Propositon 6.2. *All edges in a besom have the same color for any possible edge coloring.*

This condition is necessary, because a besom represents one time unit of a group operation a_{jl}, for which all processors in G_l have to perform at the same time job J_j.

6.2 The algorithm

The procedure for each edge is presented here in pseudocode containing two main steps. We use the notation e for edge.

INPUT: A bipartite graph presenting all processing times;

OUTPUT: An t^*-edge coloring which represent the timetable;

Algorithm:

IF ($e \neq$ besom)
make an proper t-edge coloration where $t = W$;
ENDIF

IF ($e =$ besom)
color x, y;
take $x = W$;

IF ($min_{y \preccurlyeq x} y$ is possible for e)
color e with y;

ELSE color e with $x + 1$;

$x = x + 1$;
ENDIF

Step one can be done using any existing edge coloring algorithm satisfying the König's Theorem 2.2. That the algorithm works also for the second step, we have to give an additional information, which say us if the edge is included in a besom or not.

The final t^*-edge coloring ($t^* = x$) represent a timetable with makespan t^* and each color represents one time unit interval of the timetable.

6.2.1 The schedule

The schedule obtained by this algorithm satisfies the following theorem:

Theorem 6.1. *The distance between the solution obtained by the besom algorithm, denoted as t^B, and the workload W is given by*

$$t^B - W \leq \max_j \sum_{l=1}^{p} a_{jl}.$$

Proof. The theorem is valid for all graphs without besoms due to the König's Theorem. Until now we have $W = t^*$. The difficulties are appearing with the besoms.

Because the theorem is true for all graphs excluding besoms, we try to show, that if we introduce a besom the coloring is not increased by more then one additional color. Therefore t^* might change for three reasons. The degree is increased by one. All colors are already used and not valid for the actual edge. If W and t^* are increasing by one simultaneously, t^* changes, but in this case the difference does not change. Thus we have three cases, where t^* is changed. They are given in the next lemma.

Lemma 6.1. *If we add a besom in a graph G with a proper t^*-edge coloring, we have three possibilities:*

1. *W and t^* do not change;*

2. *W does not change, but t^* increases by one;*

3. *W increase by one and hence t^* has to increase by one depending on W;*

But the difference between W and t^* after the addition of a besom is not more then one.

Proof. The first case is obvious, because if we have no change in respect of W and t^*, then there must be at least one color left to color the new besom. Hence the besom fingers are joining other edges where each one has at most one of $t^* - 1$ colors and the besom head also joins an edge with one of these $t^* - 1$ color. Therefore, we can color the besom with at least color t^*. The difference remains the same, zero, because $W = t^*$.

In the second case, no edge of the besom joins a vertex of maximal degree. Yet all joining edges are assigned with all possible t^* colors. So we must color the edges of the besom with the color $t^* + 1$. The difference between W and $t^* := t^* + 1$ is now one.

The third case appears, if we add the besom at a place where one or more of its fingers or its head join in a vertex with maximal degree. Clearly we have to color all edges of the besom with color t^*+1. But in this case the maximal degree of G is increased by one and the difference between $W := W + 1$ and $t^* := t^* + 1$ is zero. □

Now we know that for each addition of a besom, t^* increases maximally by one. Hence for each besom and each group operation, we have to add maximally one color, so we have to increase the number of colors maximally by $\max_j a_{jl}$ for each multiprocessor G_l at the vertex corresponding to job J_j. This behavior is the same for any multiprocessor, so in the case where at most all multiprocessors are performed by job J_j, the upper bound is

$$\max_j \sum_{l=1}^{p} a_{jl}.$$

We do not have to consider the number of besom fingers in P_h, because there are only a_{jl} edges, where $P_h \in G_l$. This fact is based on the partition of the processors into multiprocessors. Hence, this number of group operations in P_h is included in the sum over all multiprocessors. □

6.2.2 Interpretation

Figure 6.3 represents Theorem 6.1. In the Figure we have a subgraph of G, which is bipartite, but to avoid excessive crossing of edges, we draw it in such a way that the jobs are on the left side, job J_j on the right side. In the middle are the processors, where the multiprocessors G_{l-1}, G_l and G_{l+1} are encircled. The transformed hyperedges E_i, which $|E_i| = 2$, are drawn as fine lines.

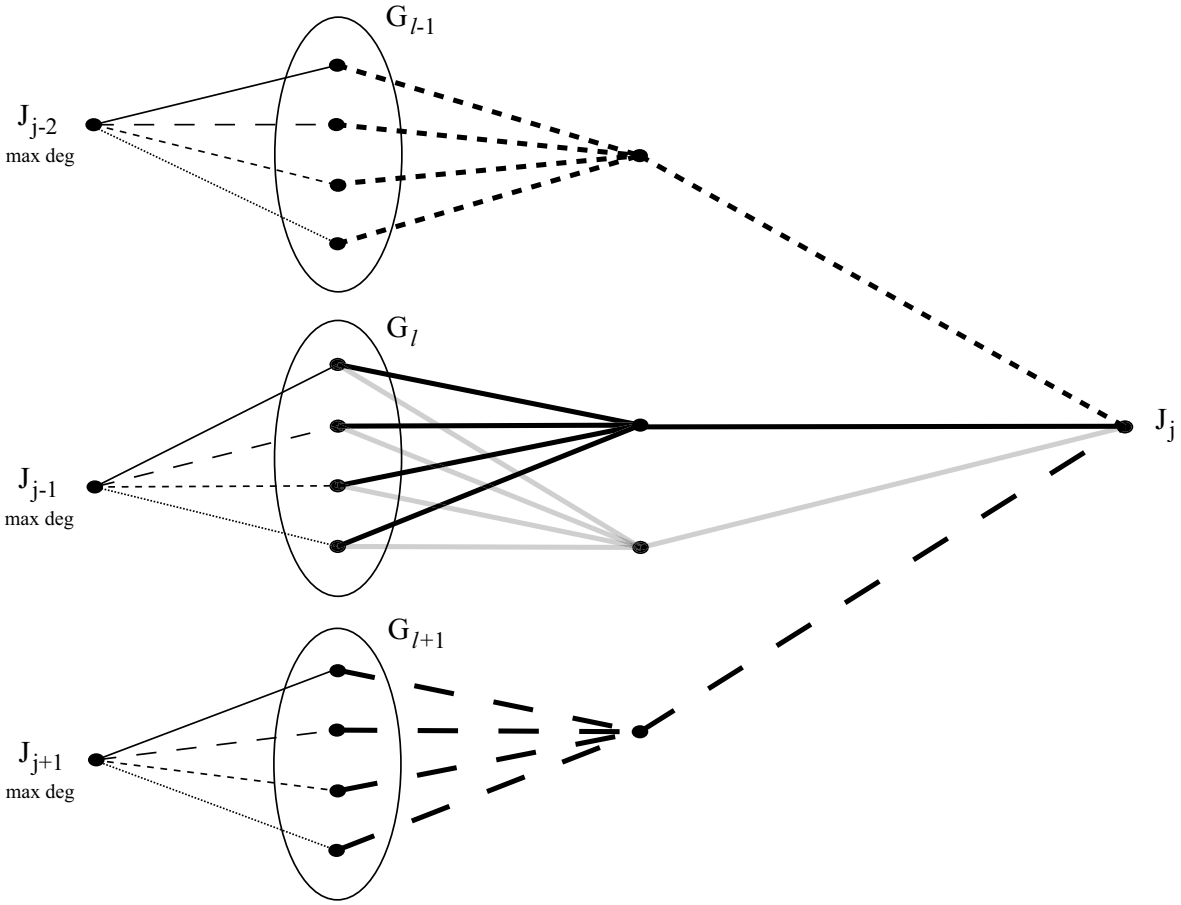

Figure 6.3: An interpretation of the result in Theorem 6.1. The besoms are the thicker lines.

Moreover the vertices J_{j-2}, J_{j-1} and J_{j+1} have maximal degree. So all colors are represented on the edges joining these vertices and the processors of G_{l-1}, G_l and G_{l+1}.To add a besom joining the vertices of G_l and vertex J_j, we need a new color (take the thick solid black line as first one). For each additional group operation between the edges of G_l and J_j we have to add a_{jl} colors for each group operation $\hat{O}_{jl}$ (take the thick solid grey line as second).

If another multiprocessor, say G_{l-1}, also joins J_j, we need another color (take the thick tiny doted line). Also, for this multiprocessor we must add the same

number of colors as the multiplicity of the group operation. We have to continue for all new besoms joining in J_j, like G_{l+1} (for this one take the thick large dotted line).

Hence, our result is satisfied, because we need at least the same number of colors as the maximal degree. Furthermore we need the number of colors equal to the number of maximal joining edges between a besom and one of the jobs, that means the $\max_j \sum_{l=1}^{p} a_{jl}$.

Remark 6.2. *The besom algorithm used for the construction of university timetables is very useful, because you do not have a teacher, who gives many group-lectures to different groups. At the EPFL as example, there we have a basic analysis course for different engineer classes. But there is one professor, who is teaching it. Furthermore, this professor normally gives at most one other lecture. For example, a lecture to one class, but never or seldom a second group-lecture to another group of classes. In this case we would have a maximal number of additional colors equal to the maximal number of given hours for any group-lectures. That means in our result, the sum over all multiprocessors, might be pointless or unnecessary.*

6.3 Examples

6.3.1 An Example showing the sharpness of the besom algorithm's upper bound

The well known Example given in 3.11 with the $\frac{7}{3}$ fractional makespan, can be represented as in Figure 6.4. The thin lines represent one time unit of an individual operation and the thick lines represent a besom, which is one time unit of group operation. Here, each operation has only one time unit.

If we apply the algorithm on this Example described in Section 6.2, we color first all thin lines, which represent the individual operations. We obtain as intermediate result the graph in Figure 6.5. The result is obvious, because if we consider only the individual operations without the besoms, we have a bipartite graph with maximal degree two. Thus, due to the König's Theorem we need two colors.

Now, if we consider the besom in the graph, the maximal degree does not change, but we must add a new color, because the edges of the besom are adjacent to both used colors. Also, both edges are contained in the same path, $P_1J_3P_3J_4P_4J_2P_2$, which has a even number of edges, although the first and the last edge has certainly different colors and we need a third color to color the besom. If we consider the result about the upper bound, we need $W = 2$ colors for the workload and one color because of $\max_j(a_{j1} + a_{j2}) = 1$.

So we have reached the upper bound. Hence we can say the given upper bound is sharp.

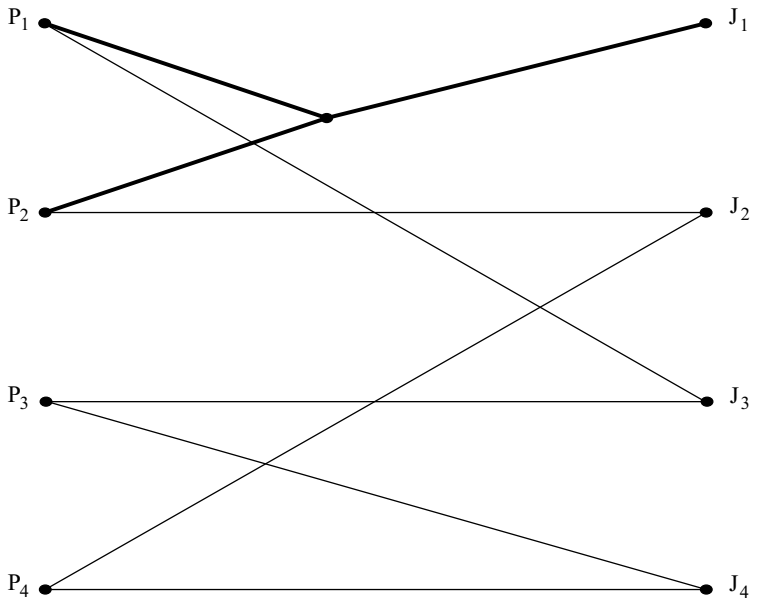

Figure 6.4: The graph of the Example presented in Figure 3.11.

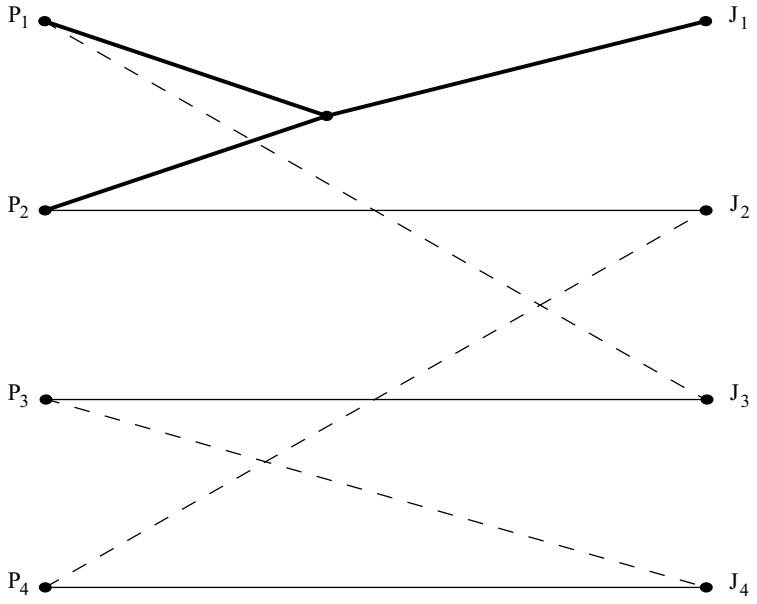

Figure 6.5: The graph with the edge coloration for the individual operations.

6.3.2 An Example where $t = W$

We obtain always a minimal timetable with makespan t, where $t = W$, if the maximal degree is at the vertex P_h, where $P_h \in G_l$ and $\sum_{j=1}^{n} a_{jl}$ is maximal. For example, we can take the exact same Example as above in Figure 6.4, because there we have P_1 or P_2 which are verifying these two conditions.

Chapter 7

Comparison of the upper bounds between our algorithms and the algorithms of Asratian and de Werra

7.1 Comparison the algorithms using graph theory

Here we compare our algorithm with the algorithms of Asratian and de Werra given in [1].

First we inform that Asratian and de Werra have also a second algorithm presented in their paper, denoted as A_2. We denote the $\frac{7}{6}$ optimality algorithm under natural assumption as A_1. But they construct a different graph. They create, on one side, a vertex for each processor and each multiprocessor. On the other side, they put a vertex for each job. Now, for each time unit of a group operation $\hat{O}_{jl}$ there is an edge between G_l and J_j. Also, for each time unit of a individual operation O_{jh} an edge joins P_h and J_j.

If we consider the Example in Figure 6.4, their graph would be drawn as in Figure 7.1.

Before we start with the examples comparing the upper bounds and makespans, we show the upper bounds in Table 7.1.

The comparison of the upper bound between A_1 and the besom algorithm is given in Subsection 7.1.1, between A_2 and the besom algorithm in Subsection 7.1.2. The discussion between the upper bounds of A_1 and A_2 is given by Asratian and de Werra in [1].

Furthermore, we compare also the makespans between the graph theoretical algorithms. The results of the comparison are given in the remark 7.1 for A_2 vs besom algorithm and in remark 7.2 for A_1 vs besom algorithm.

Algorithm	upper bound
A_1	$\max\left(\max_l(\sum_{j=1}^{n} a_{jl}), \max_h(\sum_{j=1}^{n} b_{jh}), \max_j(\sum_{l=1}^{p} a_{jl} + \sum_{h=1}^{m} b_{jh})\right)$ $+ \max\left(\max_j(\sum_{l=1}^{p} a_{jl}), \max_j(\sum_{h=1}^{m} b_{jh}), \lfloor \frac{W}{2} \rfloor\right)$
A_2	$\max\left(\max_j(\sum_{h=1}^{m} b_{jh}), \max_h(\sum_{j=1}^{n} b_{jh})\right)$ $+ \max\left(\max_j(\sum_{l=1}^{p} a_{jl}), \max_l(\sum_{j=1}^{n} a_{jl})\right)$
besom	$\max\left(\max_j w(J_j), \max_h w(P_h)\right) + \max_j(\sum_{l=1}^{p} a_{jl})$

Table 7.1: The upper bounds for the graph theoretical algorithms

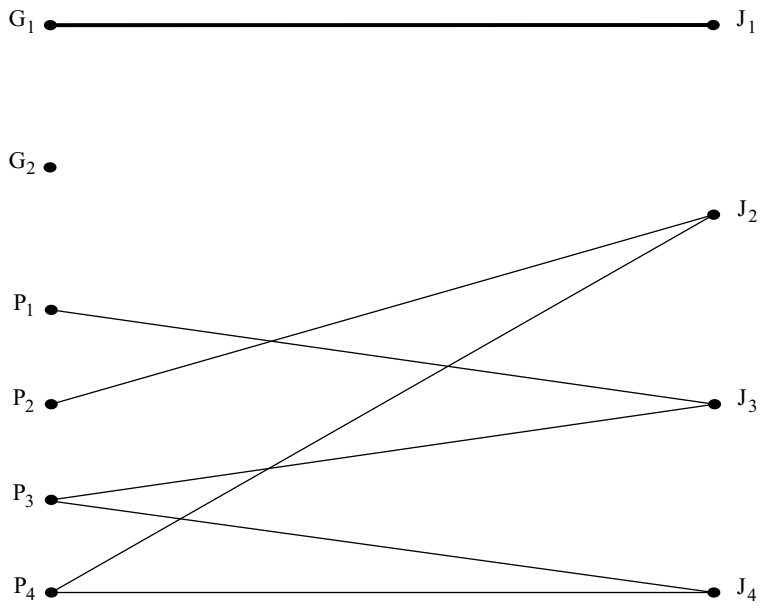

Figure 7.1: An Example of a graph drawn by the alghorithm of Asratian and de Werra.

7.1.1 Comparison between A_1 and the besom method

To show which upper bound is better, we split the question in two parts. Part one considers all examples where the maximal degree is in J_j (respective in T_j). Part two takes all remained examples.

Maximal degree in J_j: If the maximal degree is in one of the J_j, then the first max of the A_1 upper bound is equal to the workload W. Then we have equal upper bounds between the two algorithms, if the $\max_j \sum_{l=1}^{p} a_{jl}$ is the biggest term in the second max. If not, our algorithm has the better upper bound because $\max_j(\sum_{h=1}^{m} b_{jh})$ or $\lfloor \frac{W}{2} \rfloor$ is bigger then $\max_j \sum_{l=1}^{p} a_{jl}$.

Remark to the remaining examples: I examined many examples, but in no example I have seen a better upper bound for A_1.

Example 7.1. *As example we take following matrices:*

$$B = \begin{pmatrix} 1 & 0 & 1 \\ 2 & 1 & 0 \\ 0 & 2 & 2 \\ 0 & 1 & 1 \end{pmatrix}, A = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

The obtained result for the algorithm A_1 is drawn in Figure 7.2 and for our algorithm in Figure 7.3. We see the upper bound is 8 for A_1, because of $\max(2,4,5)+\max(2,4,3)=9$. *The upper bound for our algorithm is 7, 5 for the maximal workload and 2 for the maximal group operations in one of the jobs J_j. We have an 6-edge coloring for A_1 and a 5-edge coloring for our the besom method as we can see in the Figures 7.2 and 7.3 respectively. The makespans are 6 for A_1 and 5*

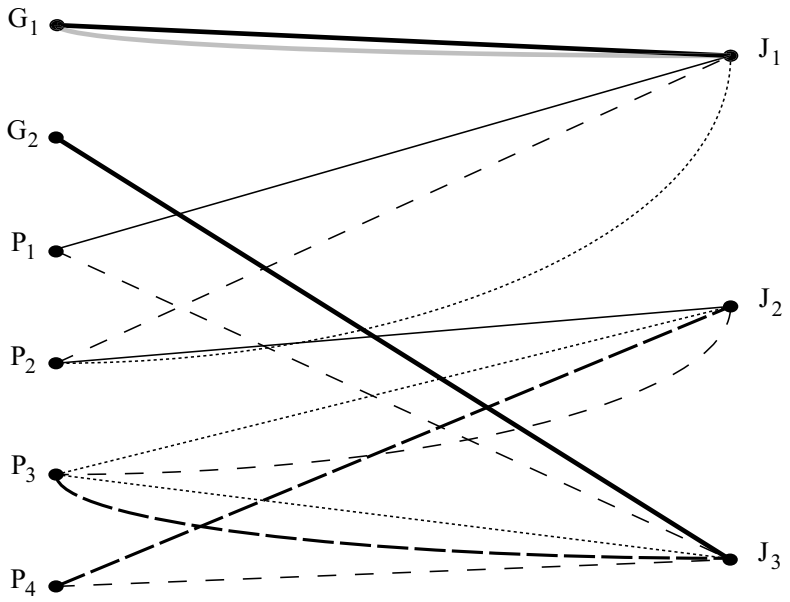

Figure 7.2: The graph drawn by the method of Asratian and De Werra.

for the besom method.

7.1.2 Comparison between A_2 and the besom method

Now, the second algorithm, A_2, colors first the individual operations with the colors $\beta_1, \ldots, \beta_u$ and the group operations with the colors $\alpha_1, \ldots, \alpha_v$. u is the maximal degree of individual operations and v is the maximal degree of group

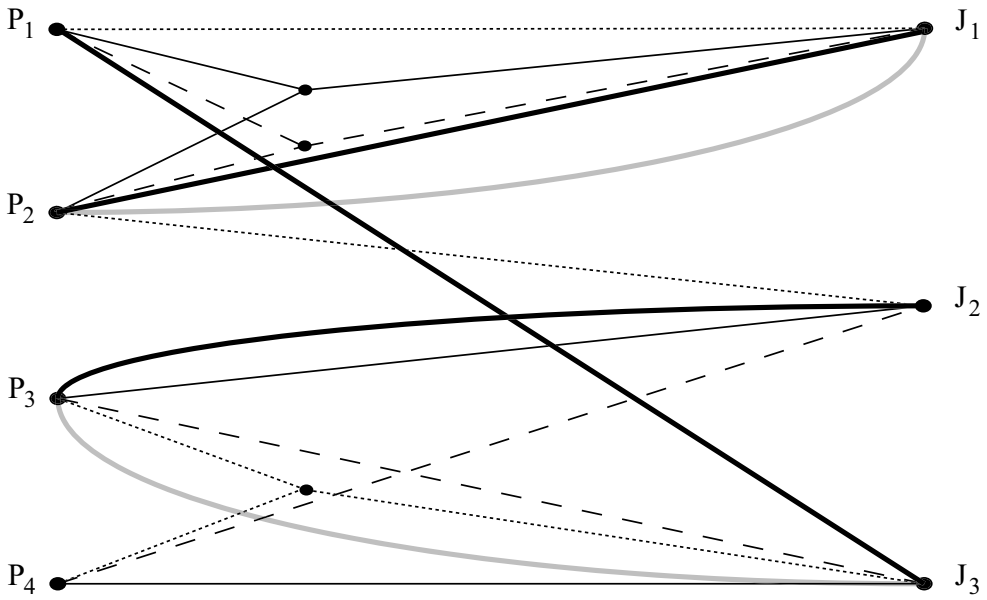

Figure 7.3: The graph drawn by our algorithm.

operations. Asratian and de Werra have shown in [1] that the upper bound for this algorithm is given by

$$\max\big(\max_{l}\sum_{j=1}^{n} a_{jl}, \max_{j}\sum_{l=1}^{p} a_{jl}\big) + \max\big(\max_{h}\sum_{j=1}^{n} b_{jh}, \max_{j}\sum_{i=1}^{p} b_{ji}\big).$$

If we consider now an Example with two processors P_1 and P_2, that together create the multiprocessor G_1. Furthermore we have two jobs, J_1 and J_2. As operations, there are $b_{11} = 1$, $b_{12} = 1$ and $a_{11} = 1$. The initial graphs are given in Figures 7.4 and 7.5. There we can see that the upper bound for our algorithm is 4, because $W = 3$ and $\max_j \sum_{l=1}^{p} a_{jl} = 1$ in J_1. The upper bound for A_2 is $\max(1,1) + \max(2,1) = 3$.

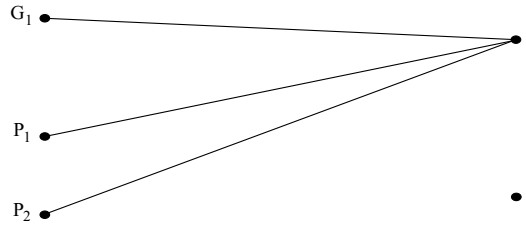

Figure 7.4: The resulting graph with the method of Asratian and de Werra.

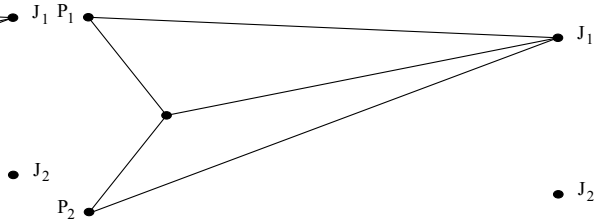

Figure 7.5: The resulting graph with the besom method.

But if we consider another Example, like the graphs in Figure 7.6 and in Figure 7.7, then we recognize that the upper bound for our algorithm is 3, because $W = 2$ and $\max_j \sum_{l=1}^{p} a_{jl} = 1$. But for A_2 we have $\max(2,1) + \max(2,2) = 4$. Hence, we can say sometimes our upper bound is better.

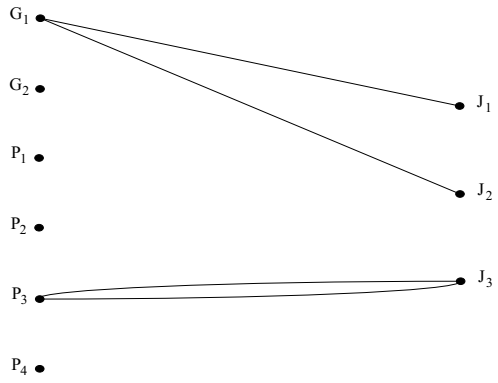

Figure 7.6: The resulting graph with the method of Asratian and de Werra.

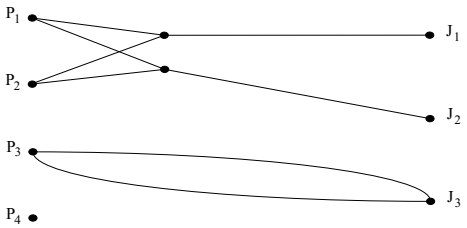

Figure 7.7: The resulting graph with the besom method.

Remark about the sharpness of the upper bounds: The Example in Figure 6.4 for our algorithm and the same Example in Figure 7.1 show the sharpness of our algorithm and algorithm A_2. In both cases we have the upper bound and the makespan both equal 3.

For the algorithm A_1 we can use the Example in Figure 7.8, where the upper bound is equal to the makespan, which is equal 2.

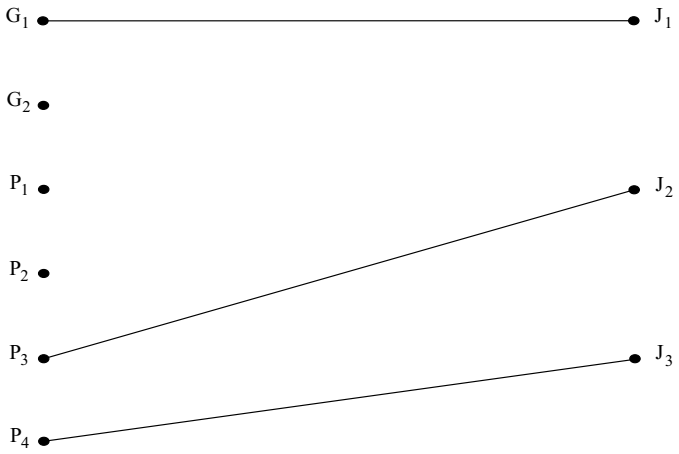

Figure 7.8: An Example showing the sharpness of the upper bound of A_1.

7.1.3 Comparison of the makespans between A_1, A_2 and the besom method

If we compare our algorithm with A_2. It is obvious that our algorithm has a shorter or equal makespan as A_2, because A_2 colors the individual operations and the group operations independently. But our algorithm tries first to color the group operations with the same colors as for individual operations before. If these colors are not possible to use, then we take a new color. Hence, A_2 obtains as best makespan the same makespan as we obtain with our algorithm, even in the case where their upper bound is smaller then our upper bound, see the Figures 7.9 and 7.10.

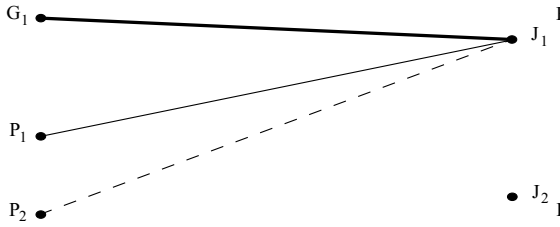

Figure 7.9: The edge coloring obtained by A_1.

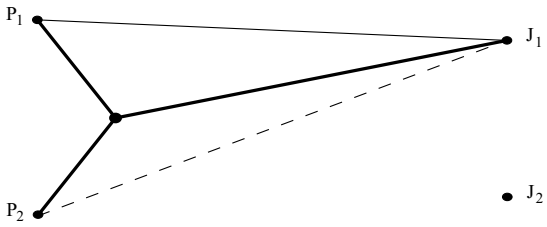

Figure 7.10: The edge coloring obtained by our algorithm.

Remark 7.1. *Because A_2 is coloring separately the individual and group operations, we know that our algorithm always produce a better or at least an equal makespan.*

To compare our algorithm with A_1, let us take following Example,

$$B = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, A = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$$

Then the edge coloring takes two colors using A_1, and there we have an upper bound equal 2. With our algorithm we obtain an edge coloration with one color and an upper bound equal 2. The results are drawn in the Figures 7.11 and 7.12.

Remark 7.2. *According to Table 7.2 and the results in Section 7.2 pictured in Table 7.3, we suppose that our algorithm always obtains a better makespan then A_1 or at least the same makespan. And it is independent, if our or their upper bound is better.*

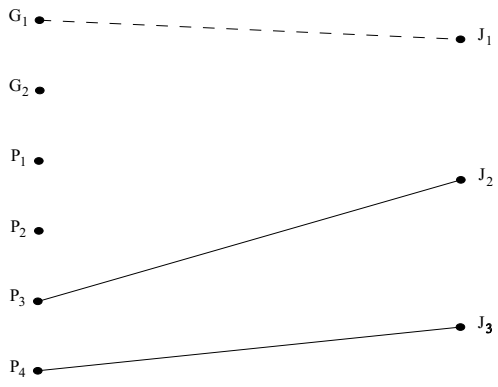

Figure 7.11: The edge coloring obtained by A_1.

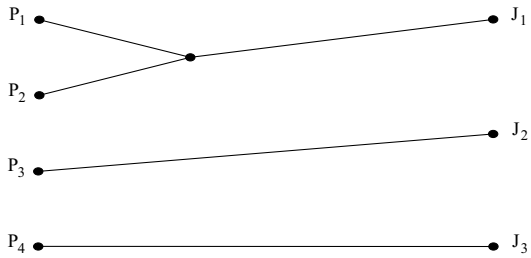

Figure 7.12: The edge coloring obtained by our algorithm.

Figures	A_1		*besom*	
	upper bound	makespan	upper bound	makespan
7.11 and 7.12	2	2	2	1
7.2 and 7.3	9	6	7	5

Table 7.2: Some comparisons between A_1 and the besom algorithm.

7.2 Compare our edge coloring algorithm with the network rounding algorithm

The Table 7.3 integrates all results discussed in the following subsections and outlines the obtained results.

7.2.1 Example with 2 multiprocessors

Consider the $\frac{7}{3}$ Example in Figure 3.11. There we obtained by the LP, the variables given in Table 7.4.

Subsection	A_1		A_2		*Besom*		(RN)		(PF)
	u.b.	t^{A_1}	u.b.	t^{A_2}	u.b.	t^B	u.b.	t^{RN}	t
7.2.1	5	3	3	3	3	3	10	3	7/3
7.2.2	12	8	8	8	10	8	28	7	7
7.2.3	86	-	104	69	75	65	120	73	61

Table 7.3: Comparisons between A_1, A_2, the besom algorithm and (RN).

$x_{11} = 1$	$y_{24} = \frac{2}{3}$
$y_{33} = \frac{2}{3}$	$y_{43} = \frac{1}{3}$
$y_{44} = \frac{1}{3}$	$a = 1$
$w = \frac{4}{3}$	$r = b = 0$

Table 7.4: The LP solution for the Example 3.11.

$y_{22} = 1$	$y_{31} = 1$
$y_{33} = 1$	$y_{44} = 1$

Table 7.5: The processing times in (d)

With these LP solutions we are able to create the rounding network, shown in Figure 7.13. The brackets represent the bounds and the resulting flow. The first number is the lower bound, the thick middle number is the flow and the last one is the upper bound.

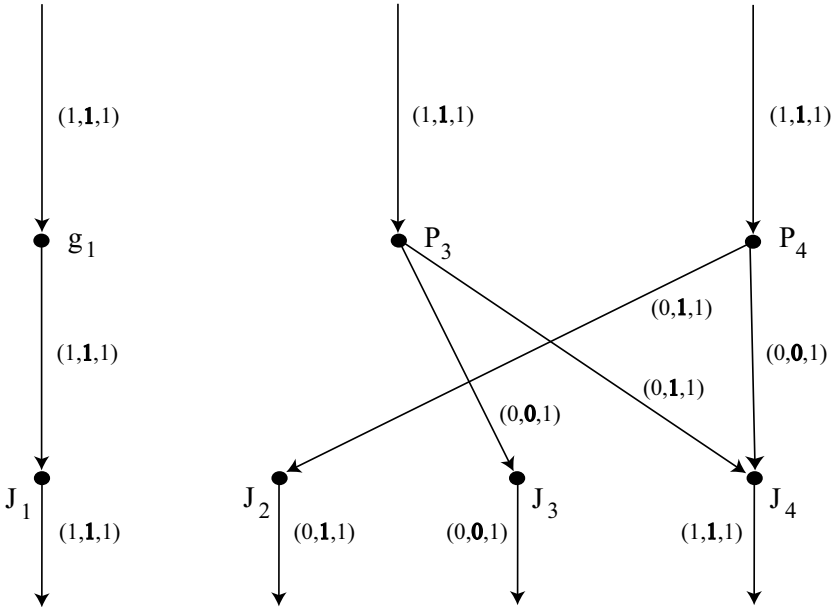

Figure 7.13: The rounding network for Example 3.11.

Then we can obtain the processing times for the interval (a). (b) and (c) are zero, due to the network. We can create the schedule for (d) with an edge coloring, as there are no group operations.

The processing times for the remaining individual operations in interval (d) are given in the Table 7.5.

There we obtain finally following schedule, drawn in Figure 7.14.

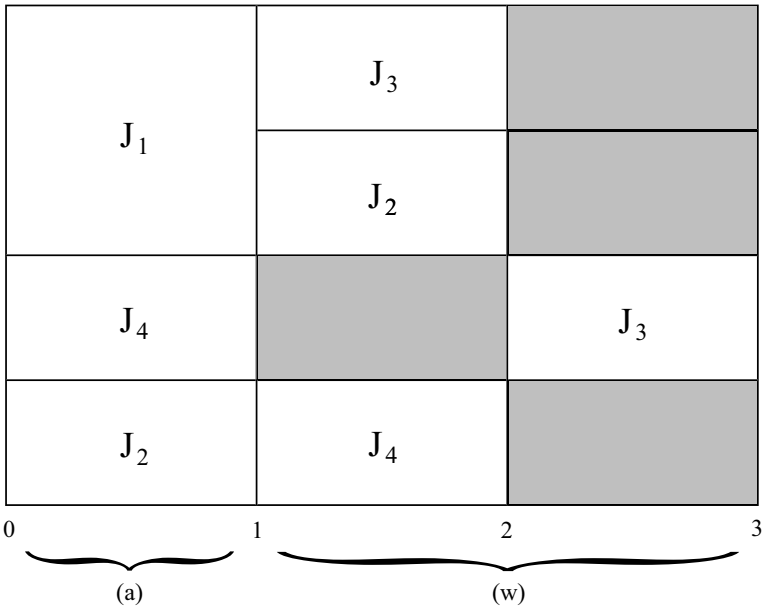

Figure 7.14: The final schedule obtained by the LP, the rounding network and the edge coloring for (w) for the Example in Figure 3.11.

For the rounding network we have a makespan equal 3, for the algorithm A_1, A_2 and the besom algorithm we obtain a makespan equal 3 too. Let us consider a bigger example.

7.2.2 Example with 3 multiprocessors

As a second Example we consider the matrices

$$B = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 2 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 2 \end{pmatrix}, A = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 2 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

and the multiprocessors $G_1 = \{P_1, P_2, P_3\}$, $G_2 = \{P_4, P_5\}$ and $G_3 = \{P_6, , P_7, P_8\}$.

We obtain a makespan equal 7 by the rounding network which is equal to the (PF) solution. For the algorithms A_1, A_2 and the besom algorithm we obtain a makespan equal 8. The result for A_1 and A_2 are shown in Figure 7.15, for the besom method in Figure 7.16 and for the network we present the values in Table 7.6. The final schedule obtained by (RN) is shown in Figure 7.17.

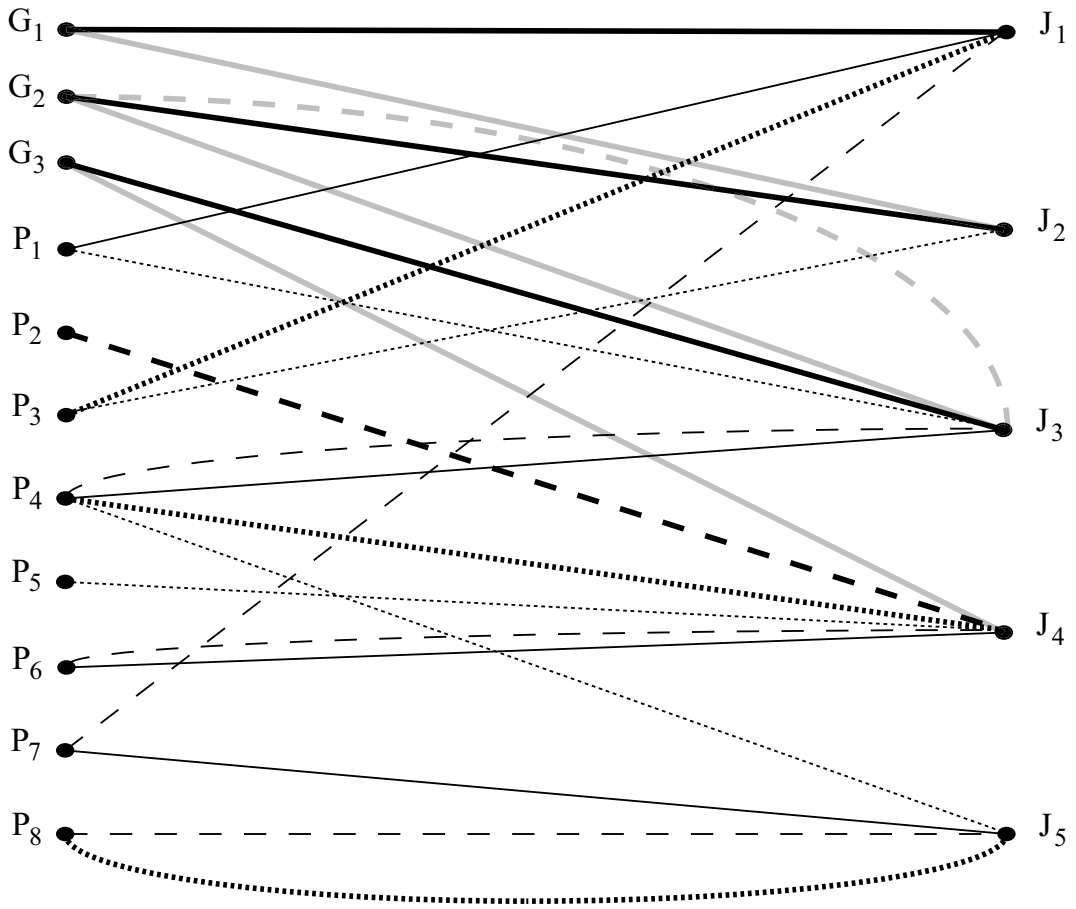

Figure 7.15: The edge coloring using A_1 or A_2.

7.2.3 Example with 4 multiprocessors

In this Example with four multiprocessors we give only the final makespan for the (RN), the besom algorithm and A_2. The Input data is given by the matrices A and B. The multiprocessors are the sets $G_1 = \{P_1, P_2, P_3\}$, $G_2 = \{P_4, \ldots, P_7\}$, $G_3 = \{P_8, P_9\}$ and $G_4 = \{P_{10}, P_{11}, P_{12}\}$.

$$B = \begin{pmatrix} 2 & 3 & 1 & 4 & 2 & 2 & 1 & 3 & 2 & 1 & 5 & 4 \\ 1 & 2 & 4 & 4 & 3 & 2 & 3 & 3 & 2 & 2 & 1 & 3 \\ 4 & 4 & 2 & 2 & 2 & 3 & 1 & 2 & 4 & 3 & 5 & 2 \\ 6 & 2 & 4 & 5 & 1 & 2 & 3 & 3 & 2 & 4 & 1 & 1 \\ 1 & 1 & 1 & 5 & 2 & 2 & 3 & 1 & 4 & 2 & 3 & 2 \\ 2 & 1 & 3 & 3 & 5 & 3 & 2 & 2 & 5 & 2 & 2 & 1 \\ 3 & 3 & 3 & 2 & 1 & 4 & 3 & 2 & 2 & 4 & 4 & 5 \\ 2 & 2 & 4 & 4 & 1 & 3 & 4 & 4 & 2 & 2 & 5 & 3 \\ 1 & 2 & 3 & 4 & 4 & 5 & 4 & 3 & 3 & 5 & 2 & 1 \\ 5 & 5 & 4 & 2 & 2 & 1 & 2 & 3 & 5 & 4 & 2 & 2 \end{pmatrix}, A = \begin{pmatrix} 2 & 2 & 1 & 4 \\ 3 & 5 & 1 & 2 \\ 4 & 1 & 1 & 3 \\ 5 & 5 & 2 & 2 \\ 2 & 2 & 1 & 1 \\ 3 & 1 & 2 & 3 \\ 4 & 4 & 1 & 4 \\ 2 & 1 & 1 & 2 \\ 3 & 1 & 3 & 2 \\ 4 & 1 & 1 & 1 \end{pmatrix}$$

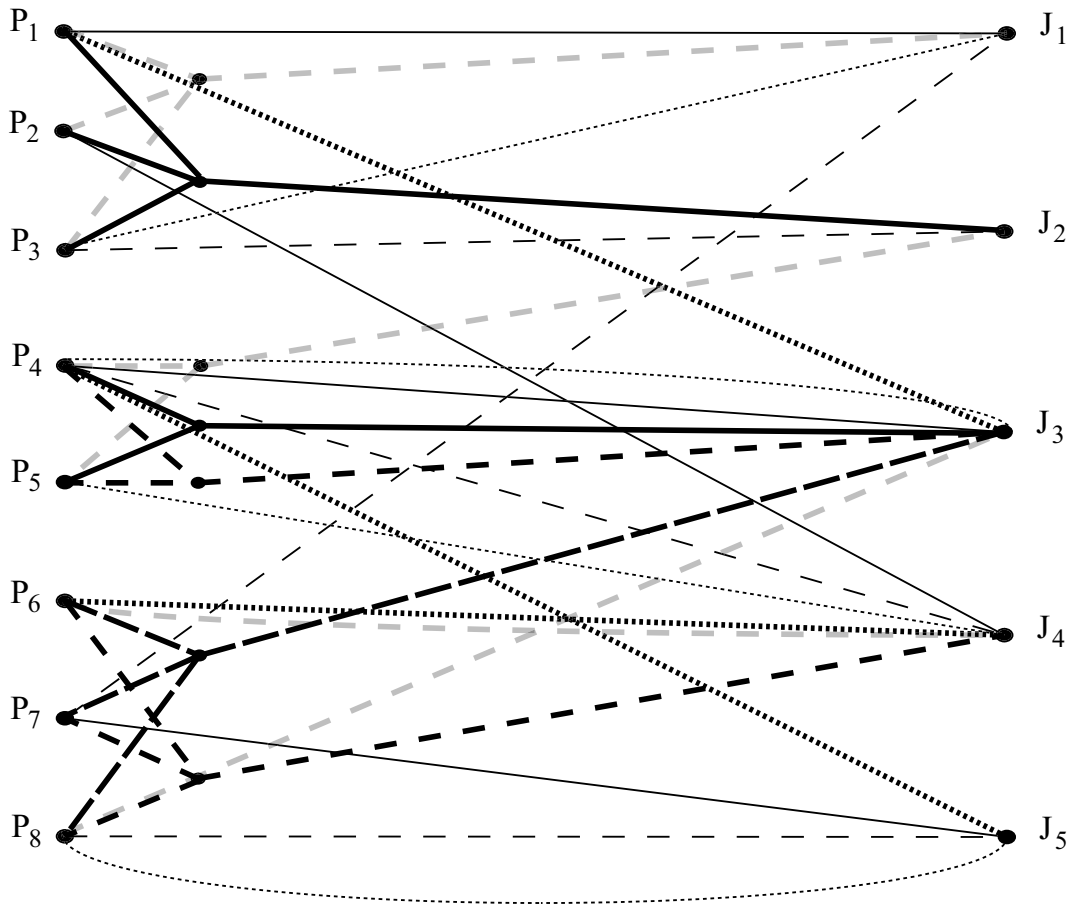

Figure 7.16: The edge coloring using the besom algorithm.

After the computation of the LP we obtained a makespan of 61 for the fractional solution. The fractional makespan is equal to the maximal workload that is on processor P_3. Then, using the (RN) we calculated a new makespan of 73. To compare this solution we took A_2 which returned us a makespan of 69 and the besom algorithm returned a makespan of 65. The upper bound for (RN) is 120 (61+59), for the besom algorithm 75 (61+15).

$y_{11}^1 = 1$	$y_{13}^1 = 1$	$y_{34}^1 = 1$	$y_{45}^1 = 1$	$y_{54}^1 = 1$
$x_{33}^1 = 1$	$x_{43}^1 = 1$	$y_{31}^2 = 1$	$y_{23}^2 = 1$	$y_{42}^2 = 1$
$y_{46}^2 = 1$	$y_{17}^2 = 1$	$y_{57}^2 = 1$	$y_{58}^2 = 1$	$x_{22}^2 = 1$
$x_{32}^2 = 1$	$y_{34}^3 = 1$	$y_{44}^3 = 1$	$y_{38}^3 = 1$	$x_{11}^3 = 1$
$x_{21}^3 = 1$	$a_1 = 0$	$a_2 = 2$	$b_1 = 0$	$b_2 = 3$
$c_1 = 0$	$c_2 = 2$	$w = 0$	$r = 0$	

Table 7.6: The LP solution for the second Example.

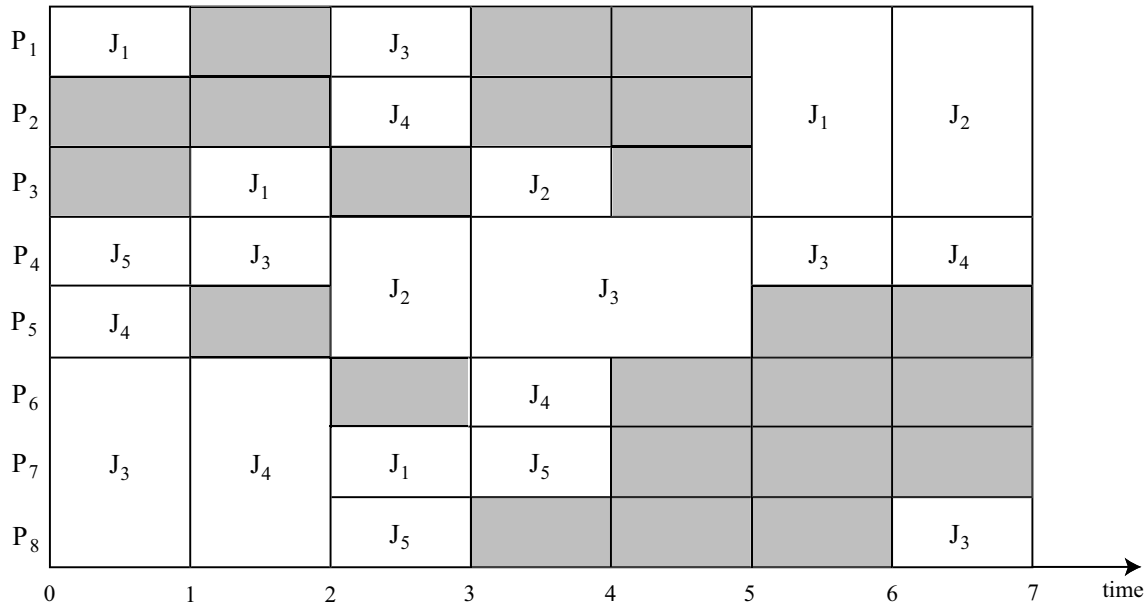

Figure 7.17: The final schedule.

Chapter 8

Conclusion

In this thesis we have shown following results:

- the LP formulation for open shops with $p = 1, 4$ and arbitrary number of multiprocessors;
- the LP concept for the resolution of open shops with any fixed number of multiprocessors;
- a graph theoretical algorithm for open shops with $p = 1$;
- a rounding network (RN) whose (PI) solution t^{RN} for any fixed number of multiprocessors p in the university model is at a constant distance from the optimal (PF) solution t,

$$t^{RN} - t \leq 2p \cdot (2^{p-1} - 1) + 3;$$

- a rounding network (RN) whose (PI) solution t^{RN} for any fixed number of multiprocessors p in the professor-lecturer model is at a constant distance form the optimal (PF) solution t,

$$t^{RN} - t \leq (p + 1)(2^{p-1} - 1) + 2;$$

- a graph theoretical algorithm, the besom algorithm, whose (PI) solution t^B for any fixed number of multiprocessors p in the university model, has an upper bound

$$t^B \leq W + \max_j \sum_{l=1}^{p} a_{jl};$$

- some examples comparing the network rounding (RN), the besom algorithm with two algorithms given by Asratian and de Werra in [1];

The main result is that the difference between optimal fractional and the integer makespan is constant for any fixed number of p multiprocessors. This result could be achieved due to the rounding network. An idea based, by Kubiak in [13], on the case with two multiprocessors. Furthermore, we have demonstrated using the besom algorithm that the difference between the maximal workload and the fractional makespan is not constant for any fixed p.

Following tasks or questions possibly may be discussed in future research:

- the crucial question is still to determine whether it is possible to find an optimal fractional solution for arbitrary number of multiprocessors in polynomial time;
- to program the (RN) and the besom algorithm and compare their quality of solutions and performance;
- to model more features of the problem such as double lectures, no long breaks during the day, availability of teachers, etc.

Appendix A

The bounds of (RN)

A.1 The bounds given by μ and λ

$$\mu_1 = (\lfloor \sum_{i=1}^{3} \sum_{j=1}^{n} x_{j1}^{(i)} \rfloor, \lceil \sum_{i=1}^{3} \sum_{j=1}^{n} x_{j1}^{(i)} \rceil)$$

$$\mu_2 = (\lfloor \sum_{i=1}^{3} \sum_{j=1}^{n} x_{j2}^{(i)} \rfloor, \lceil \sum_{i=1}^{3} \sum_{j=1}^{n} x_{j2}^{(i)} \rceil)$$

$$\mu_3 = (\lfloor \sum_{i=1}^{3} \sum_{j=1}^{n} x_{j3}^{(i)} \rfloor, \lceil \sum_{i=1}^{3} \sum_{j=1}^{n} x_{j3}^{(i)} \rceil)$$

$$\lambda_{h_1} = (\lfloor \sum_{i=1}^{3} \sum_{j=1}^{n} y_{jh_1}^{(i)} \rfloor, \lceil \sum_{i=1}^{3} \sum_{j=1}^{n} y_{jh_1}^{(i)} \rceil), \ \forall h_1 \in G_1$$

$$\lambda_{h_2} = (\lfloor \sum_{i=1}^{3} \sum_{j=1}^{n} y_{jh_2}^{(i)} \rfloor, \lceil \sum_{i=1}^{3} \sum_{j=1}^{n} y_{jh_2}^{(i)} \rceil), \ \forall h_2 \in G_2$$

$$\lambda_{h_3} = (\lfloor \sum_{i=1}^{3} \sum_{j=1}^{n} y_{jh_3}^{(i)} \rfloor, \lceil \sum_{i=1}^{3} \sum_{j=1}^{n} y_{jh_3}^{(i)} \rceil), \ \forall h_3 \in G_3$$

A.2 The bounds given by α and β

$$\alpha_1^1 = (\lfloor \sum_{j=1}^{n} x_{j1}^{(1)} \rfloor, \lceil \sum_{j=1}^{n} x_{j1}^{(1)} \rceil) \qquad \beta_1^1 = (\lfloor \sum_{j=1}^{n} y_{jh_3}^{(1)} \rfloor, \lceil \sum_{j=1}^{n} y_{jh_3}^{(1)} \rceil)$$

$$\alpha_2^1 = (\lfloor \sum_{j=1}^{n} x_{j2}^{(1)} \rfloor, \lceil \sum_{j=1}^{n} x_{j2}^{(1)} \rceil) \qquad \beta_2^1 = (\lfloor \sum_{j=1}^{n} y_{jh_1}^{(1)} \rfloor, \lceil \sum_{j=1}^{n} y_{jh_1}^{(1)} \rceil)$$

$$\alpha_3^1 = (\lfloor\sum_{j=1}^{n} x_{j3}^{(1)}\rfloor, \lceil\sum_{j=1}^{n} x_{j3}^{(1)}\rceil) \qquad \beta_3^1 = (\lfloor\sum_{j=1}^{n} y_{jh_2}^{(1)}\rfloor, \lceil\sum_{j=1}^{n} y_{jh_2}^{(1)}\rceil)$$
$$\alpha_1^2 = (\lfloor\sum_{j=1}^{n} x_{j1}^{(2)}\rfloor, \lceil\sum_{j=1}^{n} x_{j1}^{(2)}\rceil) \qquad \beta_1^2 = (\lfloor\sum_{j=1}^{n} y_{jh_2}^{(2)}\rfloor, \lceil\sum_{j=1}^{n} y_{jh_2}^{(2)}\rceil)$$
$$\alpha_2^2 = (\lfloor\sum_{j=1}^{n} x_{j3}^{(2)}\rfloor, \lceil\sum_{j=1}^{n} x_{j3}^{(2)}\rceil) \qquad \beta_2^2 = (\lfloor\sum_{j=1}^{n} y_{jh_1}^{(2)}\rfloor, \lceil\sum_{j=1}^{n} y_{jh_1}^{(2)}\rceil)$$
$$\alpha_3^2 = (\lfloor\sum_{j=1}^{n} x_{j2}^{(2)}\rfloor, \lceil\sum_{j=1}^{n} x_{j2}^{(2)}\rceil) \qquad \beta_3^2 = (\lfloor\sum_{j=1}^{n} y_{jh_3}^{(2)}\rfloor, \lceil\sum_{j=1}^{n} y_{jh_3}^{(2)}\rceil)$$
$$\alpha_1^3 = (\lfloor\sum_{j=1}^{n} x_{j2}^{(3)}\rfloor, \lceil\sum_{j=1}^{n} x_{j2}^{(3)}\rceil) \qquad \beta_1^3 = (\lfloor\sum_{j=1}^{n} y_{jh_1}^{(3)}\rfloor, \lceil\sum_{j=1}^{n} y_{jh_1}^{(3)}\rceil)$$
$$\alpha_2^3 = (\lfloor\sum_{j=1}^{n} x_{j3}^{(3)}\rfloor, \lceil\sum_{j=1}^{n} x_{j3}^{(3)}\rceil) \qquad \beta_2^3 = (\lfloor\sum_{j=1}^{n} y_{jh_2}^{(3)}\rfloor, \lceil\sum_{j=1}^{n} y_{jh_2}^{(3)}\rceil)$$
$$\alpha_3^3 = (\lfloor\sum_{j=1}^{n} x_{j1}^{(3)}\rfloor, \lceil\sum_{j=1}^{n} x_{j1}^{(3)}\rceil) \qquad \beta_3^3 = (\lfloor\sum_{j=1}^{n} y_{jh_3}^{(3)}\rfloor, \lceil\sum_{j=1}^{n} y_{jh_3}^{(3)}\rceil)$$

A.3 The bounds given by θ

$$\theta_{11}^1 = (\lfloor x_{11}^{(1)}\rfloor, \lceil x_{11}^{(1)}\rceil), \quad \theta_{j1}^1 = (\lfloor x_{j1}^{(1)}\rfloor, \lceil x_{j1}^{(1)}\rceil), \quad \theta_{n1}^1 = (\lfloor x_{n1}^{(1)}\rfloor, \lceil x_{n1}^{(1)}\rceil)$$
$$\theta_{12}^1 = (\lfloor x_{12}^{(1)}\rfloor, \lceil x_{12}^{(1)}\rceil), \quad \theta_{j2}^1 = (\lfloor x_{j2}^{(1)}\rfloor, \lceil x_{j2}^{(1)}\rceil), \quad \theta_{n2}^1 = (\lfloor x_{n2}^{(1)}\rfloor, \lceil x_{n2}^{(1)}\rceil)$$
$$\theta_{13}^1 = (\lfloor x_{13}^{(1)}\rfloor, \lceil x_{j3}^{(1)}\rceil), \quad \theta_{j3}^1 = (\lfloor x_{j3}^{(1)}\rfloor, \lceil x_{j3}^{(1)}\rceil), \quad \theta_{n3}^1 = (\lfloor x_{n3}^{(1)}\rfloor, \lceil x_{n3}^{(1)}\rceil)$$
$$\theta_{11}^2 = (\lfloor x_{11}^{(2)}\rfloor, \lceil x_{11}^{(2)}\rceil), \quad \theta_{j1}^2 = (\lfloor x_{j1}^{(2)}\rfloor, \lceil x_{j1}^{(2)}\rceil), \quad \theta_{n1}^2 = (\lfloor x_{n1}^{(2)}\rfloor, \lceil x_{n1}^{(2)}\rceil)$$
$$\theta_{13}^2 = (\lfloor x_{13}^{(2)}\rfloor, \lceil x_{13}^{(2)}\rceil), \quad \theta_{j3}^2 = (\lfloor x_{j3}^{(2)}\rfloor, \lceil x_{j3}^{(2)}\rceil), \quad \theta_{n3}^2 = (\lfloor x_{n3}^{(2)}\rfloor, \lceil x_{n3}^{(2)}\rceil)$$
$$\theta_{12}^2 = (\lfloor x_{13}^{(2)}\rfloor, \lceil x_{13}^{(2)}\rceil), \quad \theta_{j2}^2 = (\lfloor x_{j3}^{(2)}\rfloor, \lceil x_{11}^{()}\rceil), \quad \theta_{n2}^2 = (\lfloor x_{n3}^{(2)}\rfloor, \lceil x_{n3}^{(2)}\rceil)$$
$$\theta_{12}^3 = (\lfloor x_{12}^{(3)}\rfloor, \lceil x_{12}^{(3)}\rceil), \quad \theta_{j2}^3 = (\lfloor x_{j2}^{(3)}\rfloor, \lceil x_{j2}^{(3)}\rceil), \quad \theta_{n2}^3 = (\lfloor x_{n2}^{(3)}\rfloor, \lceil x_{n2}^{(3)}\rceil)$$
$$\theta_{13}^3 = (\lfloor x_{11}^{(3)}\rfloor, \lceil x_{11}^{(3)}\rceil), \quad \theta_{j3}^3 = (\lfloor x_{j3}^{(3)}\rfloor, \lceil x_{j3}^{(3)}\rceil), \quad \theta_{n3}^3 = (\lfloor x_{n3}^{(3)}\rfloor, \lceil x_{n3}^{(3)}\rceil)$$
$$\theta_{11}^3 = (\lfloor x_{11}^{(3)}\rfloor, \lceil x_{11}^{(3)}\rceil), \quad \theta_{j1}^3 = (\lfloor x_{j1}^{(3)}\rfloor, \lceil x_{j1}^{(3)}\rceil), \quad \theta_{n3}^3 = (\lfloor x_{n1}^{(3)}\rfloor, \lceil x_{n1}^{(3)}\rceil)$$

A.4 The bounds given by η

$$
\begin{array}{lll}
\eta^1_{1h''} = (\lfloor y^{(1)}_{1h_3} \rfloor, \lceil y^{(1)}_{1h_3} \rceil), & \eta^1_{jh''} = (\lfloor y^{(1)}_{jh_3} \rfloor, \lceil y^{(1)}_{jh_3} \rceil), & \eta^1_{nh''} = (\lfloor y^{(1)}_{nh_3} \rfloor, \lceil y^{(1)}_{nh_3} \rceil) \\
\eta^1_{1h} = (\lfloor y^{(1)}_{1h_3} \rfloor, \lceil y^{(1)}_{1h_1} \rceil), & \eta^1_{jh} = (\lfloor y^{(1)}_{jh_1} \rfloor, \lceil y^{(1)}_{jh_1} \rceil), & \eta^1_{nh} = (\lfloor y^{(1)}_{nh_1} \rfloor, \lceil y^{(1)}_{nh_1} \rceil) \\
\eta^1_{1h'} = (\lfloor y^{(1)}_{1h_2} \rfloor, \lceil y^{(1)}_{1h_2} \rceil), & \eta^1_{jh'} = (\lfloor y^{(1)}_{jh_2} \rfloor, \lceil y^{(1)}_{jh_2} \rceil), & \eta^1_{nh'} = (\lfloor y^{(1)}_{nh_2} \rfloor, \lceil y^{(1)}_{nh_2} \rceil) \\
\eta^2_{1h'} = (\lfloor y^{(2)}_{1h_2} \rfloor, \lceil y^{(2)}_{1h_2} \rceil), & \eta^2_{jh'} = (\lfloor y^{(2)}_{jh_2} \rfloor, \lceil y^{(2)}_{jh_2} \rceil), & \eta^2_{nh'} = (\lfloor y^{(2)}_{nh_2} \rfloor, \lceil y^{(2)}_{nh_2} \rceil) \\
\eta^2_{1h} = (\lfloor y^{(2)}_{1h!} \rfloor, \lceil y^{(2)}_{1h_1} \rceil), & \eta^2_{jh} = (\lfloor y^{(2)}_{jh_1} \rfloor, \lceil y^{(2)}_{jh_1} \rceil), & \eta^2_{nh} = (\lfloor y^{(2)}_{nh_1} \rfloor, \lceil y^{(2)}_{nh_1} \rceil) \\
\eta^2_{1h''} = (\lfloor y^{(2)}_{1h_3} \rfloor, \lceil y^{(2)}_{1h_3} \rceil), & \eta^2_{jh''} = (\lfloor y^{(2)}_{jh_3} \rfloor, \lceil y^{()}_{jh_3} \rceil), & \eta^2_{nh''} = (\lfloor y^{(2)}_{nh_3} \rfloor, \lceil y^{(2)}_{nh_3} \rceil) \\
\eta^3_{1h} = (\lfloor y^{(3)}_{1h_1} \rfloor, \lceil y^{(3)}_{1h_1} \rceil), & \eta^3_{jh} = (\lfloor y^{(3)}_{jh_1} \rfloor, \lceil y^{(3)}_{jh_1} \rceil), & \eta^3_{nh} = (\lfloor y^{(3)}_{nh_1} \rfloor, \lceil y^{(3)}_{nh_1} \rceil) \\
\eta^3_{1h'} = (\lfloor y^{(3)}_{1h_2} \rfloor, \lceil y^{(3)}_{1h_2} \rceil), & \eta^3_{jh'} = (\lfloor y^{(3)}_{jh_2} \rfloor, \lceil y^{(3)}_{jh_2} \rceil), & \eta^3_{nh'} = (\lfloor y^{(3)}_{nh_2} \rfloor, \lceil y^{(3)}_{nh_2} \rceil) \\
\eta^3_{1h''} = (\lfloor y^{(3)}_{1h_3} \rfloor, \lceil y^{(3)}_{1h_3} \rceil), & \eta^3_{jh''} = (\lfloor y^{(3)}_{jh_3} \rfloor, \lceil y^{(3)}_{jh_3} \rceil), & \eta^3_{nh''} = (\lfloor y^{(3)}_{nh_3} \rfloor, \lceil y^{(3)}_{nh_3} \rceil)
\end{array}
$$

A.5 The bounds given by σ

$$
\begin{aligned}
\sigma^1_1 &= (\lfloor x^{(1)}_{11} + x^{(1)}_{12} + \sum_{h_3 \in G_3} y^{(1)}_{1h_3} \rfloor, \lceil x^{(1)}_{11} + x^{(1)}_{12} + \sum_{h_3 \in G_3} y^{(1)}_{1h_3} \rceil) \\
\sigma^2_1 &= (\lfloor x^{(1)}_{13} + \sum_{h_1 \in G_1} y^{(1)}_{1h_1} + \sum_{h_2 \in G_2} y^{(1)}_{1h_2} \rfloor, \lceil x^{(1)}_{13} + \sum_{h_1 \in G_1} y^{(1)}_{1h_1} + \sum_{h_2 \in G_2} y^{(1)}_{1h_2} \rceil) \\
\sigma^3_1 &= (\lfloor x^{(2)}_{11} + x^{(2)}_{13} + \sum_{h_2 \in G_2} y^{(2)}_{1h_2} \rfloor, \lceil x^{(2)}_{11} + x^{(2)}_{13} + \sum_{h_2 \in G_2} y^{(2)}_{1h_2} \rceil) \\
\sigma^4_1 &= (\lfloor x^{(2)}_{12} + \sum_{h_1 \in G_1} y^{(2)}_{1h_1} + \sum_{h_3 \in G_3} y^{(2)}_{1h_3} \rfloor, \lceil x^{(2)}_{12} + \sum_{h_1 \in G_1} y^{(2)}_{1h_1} + \sum_{h_3 \in G_3} y^{(2)}_{1h_3} \rceil) \\
\sigma^1_j &= (\lfloor x^{(1)}_{j1} + x^{(1)}_{j2} + \sum_{h_3 \in G_3} y^{(1)}_{jh_3} \rfloor, \lceil x^{(1)}_{j1} + x^{(1)}_{j2} + \sum_{h_3 \in G_3} y^{(1)}_{jh_3} \rceil) \\
\sigma^2_j &= (\lfloor x^{(1)}_{j3} + \sum_{h_1 \in G_1} y^{(1)}_{jh_1} + \sum_{h_3 \in G_3} y^{(1)}_{jh_3} \rfloor, \lceil x^{(1)}_{j3} + \sum_{h_1 \in G_1} y^{(1)}_{jh_1} + \sum_{h_3 \in G_3} y^{(1)}_{jh_3} \rceil) \\
\sigma^3_j &= (\lfloor x^{(2)}_{j1} + x^{(2)}_{j3} + \sum_{h_2 \in G_2} y^{(2)}_{jh_2} \rfloor, \lceil x^{(2)}_{j1} + x^{(2)}_{j3} + \sum_{h_2 \in G_2} y^{(2)}_{jh_2} \rceil) \\
\sigma^4_j &= (\lfloor x^{(2)}_{j2} + \sum_{h_1 \in G_1} y^{(2)}_{jh_1} + \sum_{h_3 \in G_3} y^{(2)}_{jh_3} \rfloor, \lceil x^{(2)}_{j2} + \sum_{h_1 \in G_1} y^{(2)}_{jh_1} + \sum_{h_3 \in G_3} y^{(2)}_{jh_3} \rceil)
\end{aligned}
$$

$$
\begin{aligned}
\sigma_j^5 &= (\lfloor x_{j2}^{(3)} + x_{j3}^{(3)} + \sum_{h_1 \in G_1} y_{jh_1}^{(3)} \rfloor, \lceil x_{j2}^{(3)} + x_{j3}^{(3)} + \sum_{h_1 \in G_1} y_{jh_1}^{(3)} \rceil) \\
\sigma_j^6 &= (\lfloor x_{j1}^{(3)} + \sum_{h_2 \in G_2} y_{jh_2}^{(3)} + \sum_{h_3 \in G_3} y_{jh_3}^{(3)} \rfloor, \lceil x_{j1}^{(3)} + \sum_{h_2 \in G_2} y_{jh_2}^{(3)} + \sum_{h_3 \in G_3} y_{jh_3}^{(3)} \rceil) \\
\sigma_n^3 &= (\lfloor x_{n1}^{(2)} + x_{n3}^{(2)} + \sum_{h_2 \in G_2} y_{nh_2}^{(2)} \rfloor, \lceil x_{n1}^{(2)} + x_{n3}^{(2)} + \sum_{h_2 \in G_2} y_{nh_2}^{(2)} \rceil) \\
\sigma_n^4 &= (\lfloor x_{n2}^{(2)} + \sum_{h_1 \in G_1} y_{nh_1}^{(2)} + \sum_{h_3 \in G_3} y_{nh_3}^{(2)} \rfloor, \lceil x_{n2}^{(2)} + \sum_{h_1 \in G_1} y_{nh_1}^{(2)} + \sum_{h_3 \in G_3} y_{nh_3}^{(2)} \rceil) \\
\sigma_n^5 &= (\lfloor x_{n2}^{(3)} + x_{n3}^{(3)} + \sum_{h_1 \in G_1} y_{nh_1}^{(3)} \rfloor, \lceil x_{n2}^{(3)} + x_{n3}^{(3)} + \sum_{h_1 \in G_1} y_{nh_1}^{(3)} \rceil) \\
\sigma_n^6 &= (\lfloor x_{n1}^{(3)} + \sum_{h_2 \in G_2} y_{nh_2}^{(3)} + \sum_{h_3 \in G_3} y_{nh_3}^{(3)} \rfloor, \lceil x_{n1}^{(3)} + \sum_{h_2 \in G_2} y_{nh_2}^{(3)} + \sum_{h_3 \in G_3} y_{nh_3}^{(3)} \rceil)
\end{aligned}
$$

A.6 The bounds given by γ

$$
\begin{aligned}
\gamma_1 &= (\lfloor \sum_{i=1}^{3} \sum_{l=1}^{3} x_{1l}^{(i)} + \sum_{i=1}^{3} \sum_{h=1}^{m} y_{1h}^{(i)} \rfloor, \lceil \sum_{i=1}^{3} \sum_{l=1}^{3} x_{1l}^{(i)} + \sum_{i=1}^{3} \sum_{h=1}^{m} y_{1h}^{(i)} \rceil) \\
\gamma_j &= (\lfloor \sum_{i=1}^{3} \sum_{l=1}^{3} x_{jl}^{(i)} + \sum_{i=1}^{3} \sum_{h=1}^{m} y_{jh}^{(i)} \rfloor, \lceil \sum_{i=1}^{3} \sum_{l=1}^{3} x_{jl}^{(i)} + \sum_{i=1}^{3} \sum_{h=1}^{m} y_{jh}^{(i)} \rceil) \\
\gamma_n &= (\lfloor \sum_{i=1}^{3} \sum_{l=1}^{3} x_{nl}^{(i)} + \sum_{i=1}^{3} \sum_{h=1}^{m} y_{nh}^{(i)} \rfloor, \lceil \sum_{i=1}^{3} \sum_{l=1}^{3} x_{nl}^{(i)} + \sum_{i=1}^{3} \sum_{h=1}^{m} y_{nh}^{(i)} \rceil)
\end{aligned}
$$

Appendix B

Presentation abstract

Scheduling with Multiprocessors:
A rounding network algorithm with a constant upper bound

Oliver Ittig
Faculty of Business Administration
Supervisor: Prof. W. Kubiak

Abstract

Scheduling is used for a lot of different kind of processing. Imagine any type of production in industry like car assembly where we have different process as putting the wheels or coloring the chassis. A multiprocessor in the car example could be a robot with more then one arm and each arm is processing simultaneously on the car, like putting simultaneously the windshield, the rear window and the driving mirrors. Also in computer systems we use multiprocessors as in computing circuits or printing devices. Other examples are school timetables, which are of our main interest. The most common model is the class-teacher model. There are m classes $C_1, \ldots, C_m$ and n teachers $T_1, \ldots, T_n$. Each class C_h has b_{jh} lectures given by teacher T_j. Moreover each class is not be taught by more then one teacher in one hour, and no teacher must teach more then one class in each hour. The university timetable model represents the extended version including the multiprocessors. The extension is a lecture to a group of classes, take as example a basic mathematics course for all biology, chemistry and earth science students in a undergraduate program. The constraint changes into no teacher must teach more then one class or group in one hour.

In examples as computer systems fractional preemptions are the rules. But if we consider school timetables, there we must take integer preemptions. We give the formulation of a linear program (LP) to solve the examples with the fractional preemptions. To solve the examples as school timetables, we introduce a rounding network algorithm. The main result is that due to the network rounding we obtain

a integer timetable, where the difference to the makespan obtained by the LP is constant.

Furthermore using a graph edge coloring approach we show that the difference between the maximal workload and the makespan t is not constant.

Finally we compare our results with algorithms from Asratian and de Werra. Because one of their algorithm has an $\frac{7}{6}$ ratio optimality between the integer makespan and the maximal workload.

Bibliography

[1] A.S. Asratian and D. de Werra, A generalized class-teacher model for some timetabling problems, EJOR 143 (2002) 531-542.

[2] A.S. Asratian, T. Denley and R. Häggkvist, Bipartite graphs and their application, Cambridge University Press (1998).

[3] C. Berge, Graphs and Hypergraphs, North Holland (Amsterdam 1973).

[4] J. Błażewicz, K.H. Ecker, E. Pesch, G. Schmidt and J. Węglarz, Scheduling Computer and Manufacturing Processes, Springer (Heidelberg 1996).

[5] J.A. Bondy and U.S.R. Murty, Graph Theory with Applications, North-Holland (New York 1976).

[6] P. Brucker, Scheduling Algorithmes, Springer (Heidelberg 1998).

[7] P. Brucker and A. Krämer, Shop Scheduling Problems with Multiprocessor Tasks on Dedicated Processors, Osnabrücker Schriften zur Mathematik, Preprints, Heft 159 (1993).

[8] S. Cereghetti, Utilisation de la programmation linéaire pour l'ordonnancement, master thesis EPFL (2004).

[9] I. Dinur, O. Regev and C. Smyth, The Hardness of 3-Uniform Hypergraph Coloring, Institute for Advanced Study, Princeton, NJ (2002).

[10] H.N. Gabov and O. Kariv, Algorithms for edge coloring bipartite graphs and multigraphs, SIAM J. Computing 11:1, pp 117-129, (1982).

[11] M.R. Garey and D.S. Johnson, Computers and Intractability : A Guide to the Theory of NP-completeness, WH Freeman & Co. (San Francisco 1979).

[12] T. Gonzalez and S. Sahni, Open Shop Scheduling to Minimize Finish Time, Journal of the Asssociation for Computing Machinery, Vol 23, No 4, pp 665-679, (1976).

[13] W. Kubiak, Beyond binary jobs: preliminary draft, unpublished working paper (2004).

[14] E. Lawler, Combinatorial Optimization: Networks and Matroids, Holt, Rienhart and Winston (New York 1976).

[15] G.L. Nemhauser and A.H.G. Rinnooy Kan, Handbooks in Op. Res. and Management Science : Logistics of Production and Inventory, North Holland (Amsterdam 1993).

[16] D. de Werra, T.M. Liebling, J.F. Hêche, Recherche opérationnelle pour ingénieur I, PPUR (Lausanne 2003).

[17] D. de Werra, A.S. Asratian and S. Durand, Complexity of some special types of timetabling problems, Journal of Scheduling 5 (2002) 171-183.

[18] D. de Werra, T. Kis and W. Kubiak, Preemptive open shop scheduling with multiprocessors : polynomial cases and applications, (submitted for publication).

[19] D. de Werra and T. Kis, Some solvable cases of preemptive open shop scheduling with multiprocessors, EPFL ORWP 01/07 (2001).